# Sur-approximations non régulières et terminaison pour l'analyse d'accessibilité

Vivien Pelletier

Encadrée par :
Pierre Réty et Yohan Boichut

23 octobre 2017

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

UNIVERSITE D'ORLEANS

# Sommaire

# Sommaire

# Systèmes complexes

## Système complexe

Un système complexe est un ensemble constitué d'un grand nombre d'entités en interaction qui empêchent l'observateur de prévoir sa rétroaction, son comportement ou évolution par le calcul.

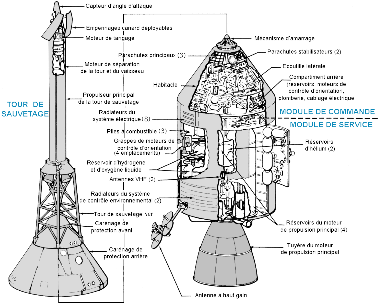
# Systèmes complexes

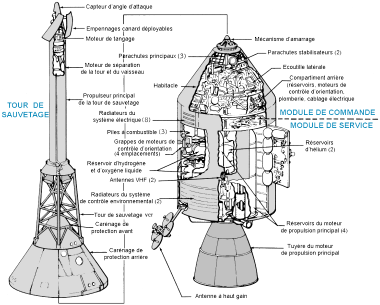
## Système complexe

Un système complexe est un ensemble constitué d'un grand nombre d'entités en interaction qui empêchent l'observateur de prévoir sa rétroaction, son comportement ou évolution par le calcul.

## Exemples

- programmes informatiques
- protocoles de sécurité
- circuits logiques


Capteur d'angle d'attaque
Empennages canard déployables
Moteur de tangage
Moteur de séparation de la tour et du vaisseau
TOUR DE SAUVETAGE
Propulseur principal de la tour de sauvetage
Tour de sauvetage
Carénage de protection avant
Carénage de protection arrière
Mécanisme d'amarrage
Parachutes stabilisateurs (2)
Parachutes principaux (3)
Ecoutille latérale
Habitacle
Compartiment arrière (réservoirs, moteurs de contrôle d'orientation, plomberie, cablage électrique
MODULE DE COMMANDE
MODULE DE SERVICE
Radiateurs du système électrique (8)
Piles à combustible (3)
Grappes de moteurs de contrôle d'orientation (4 emplacements)
Réservoirs d'hélium (2)
Réservoir d'hydrogène et d'oxygène liquide
Antennes VHF (2)
Radiateurs du système de contrôle environnemental (2)
Réservoirs du moteur de propulsion principal (4)
Tuyère du moteur de propulsion principal
Antenne à haut gain




## Génie logiciel

L'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi.

# Vérification

Conformité et fiabilité d'un système complexe ?

# Vérification

Conformité et fiabilité d'un système complexe ?

- Les tests

# Vérification

Conformité et fiabilité d'un système complexe ?

- Les tests
  - si l'ensemble d'entrées est trop grand ou infini ?

# Vérification

Conformité et fiabilité d'un système complexe ?

- Les tests
  - si l'ensemble d'entrées est trop grand ou infini ?
- Les méthodes formelles

# Vérification

Conformité et fiabilité d'un système complexe ?

- Les tests
  - si l'ensemble d'entrées est trop grand ou infini ?
- Les méthodes formelles
  - analyse statique par interprétation abstraite

# Vérification

Conformité et fiabilité d'un système complexe ?

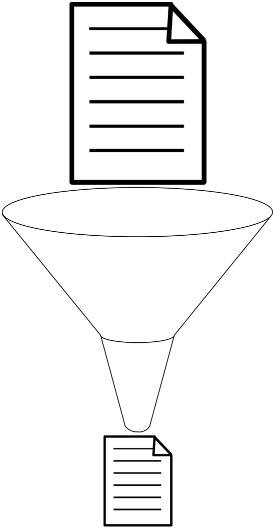
- Les tests
  - si l'ensemble d'entrées est trop grand ou infini ?
- Les méthodes formelles
  - analyse statique par interprétation abstraite
  - vérification déductive

# Vérification

Conformité et fiabilité d'un système complexe ?

- Les tests
  - si l'ensemble d'entrées est trop grand ou infini ?
- Les méthodes formelles
  - analyse statique par interprétation abstraite
  - vérification déductive
  - vérification de modèles

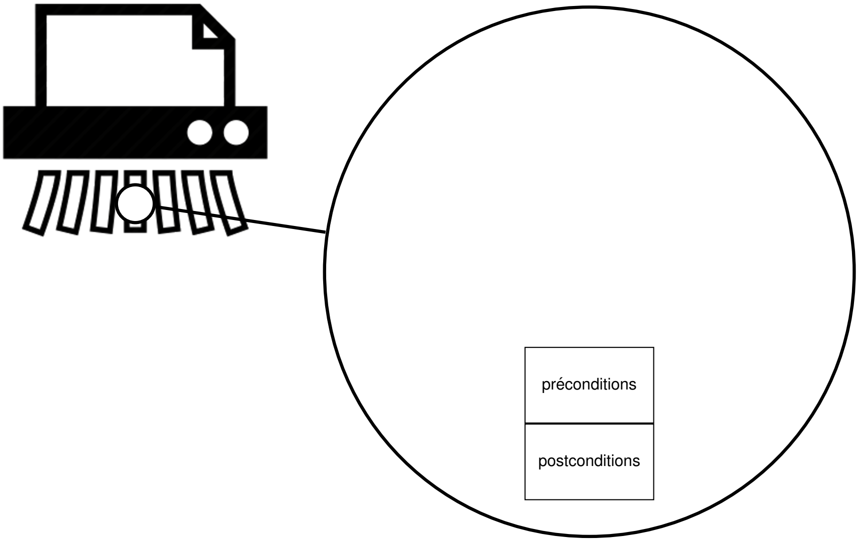
# Analyse statique par interprétation abstraite



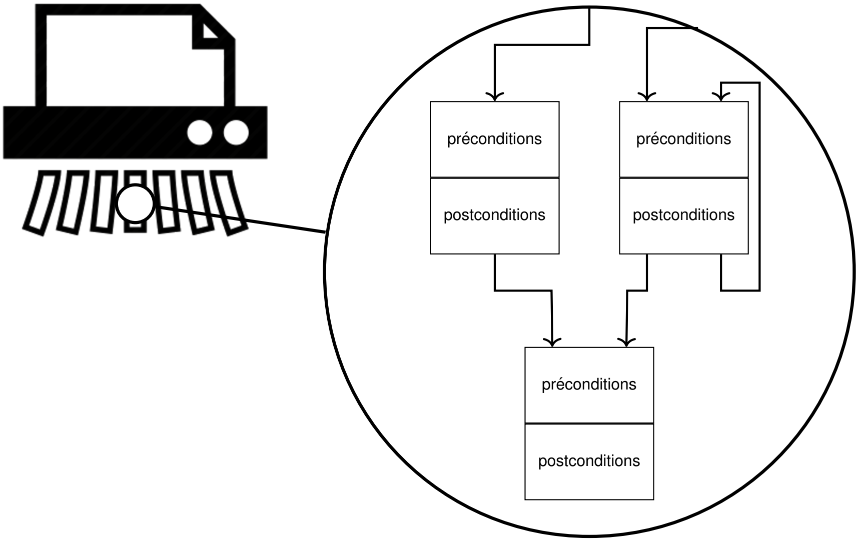
- Constat
  - trop d'informations
- Solution
  - utilisation d'abstractions
- Difficulté
  - garder suffisamment d'informations
  - mais pas trop

# Vérification déductive

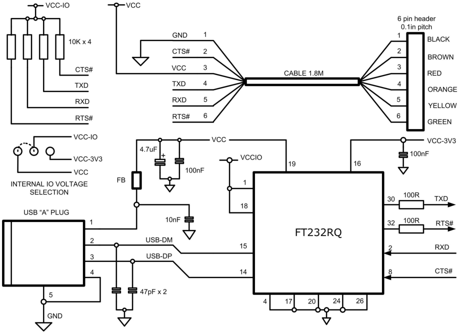
# Vérification déductive



# Vérification déductive



- Analyse exhaustive
- Représentation astucieuse



Configurations

- Ensemble des configurations accessibles

# Analyse d'accessibilité

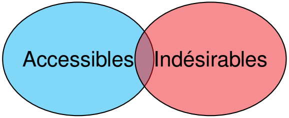
Configurations

- Ensemble des configurations accessibles
- Ensemble des configurations indésirables

# Analyse d'accessibilité

Configurations

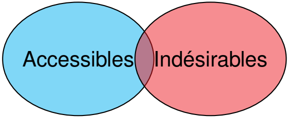
- Ensemble des configurations accessibles
- Ensemble des configurations indésirables



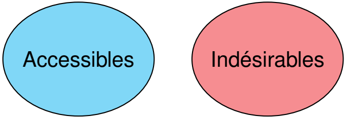

Il existe une configuration indésirable accessible

Configurations

- Ensemble des configurations accessibles
- Ensemble des configurations indésirables




Il existe une configuration indésirable accessible




Aucune configuration indésirable n'est accessible

# Calcul de l’ensemble des configurations accessibles

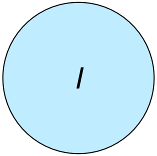
- Configurations initiales : $I$

# Calcul de l'ensemble des configurations accessibles

- Configurations initiales : $I$
- Dynamique : $R$

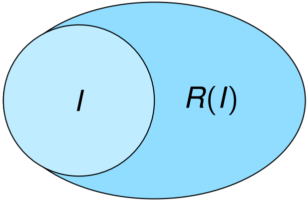
# Calcul de l’ensemble des configurations accessibles

- Configurations initiales : $I$
- Dynamique : $R$



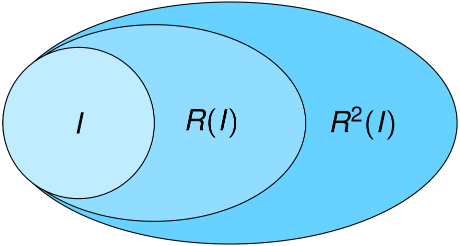
# Calcul de l'ensemble des configurations accessibles

- Configurations initiales : $I$
- Dynamique : $R$



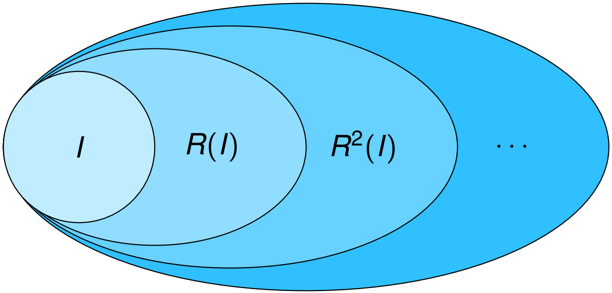
# Calcul de l'ensemble des configurations accessibles

- Configurations initiales : $I$
- Dynamique : $R$



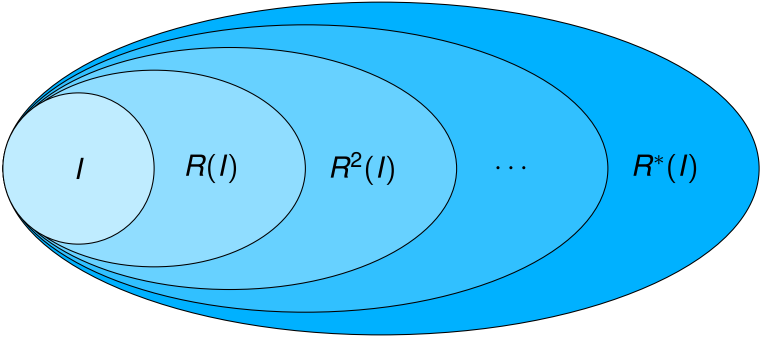
# Calcul de l'ensemble des configurations accessibles

- Configurations initiales : $I$
- Dynamique : $R$

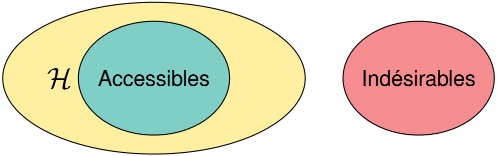


# Calcul de l'ensemble des configurations accessibles

- Configurations initiales : $I$
- Dynamique : $R$

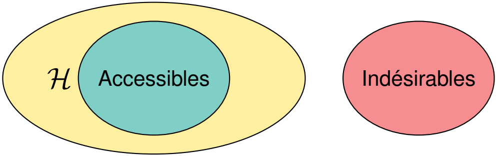


# Sur-approximations

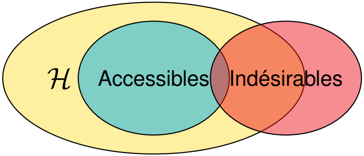



Aucune configuration indésirable accessible

# Sur-approximations

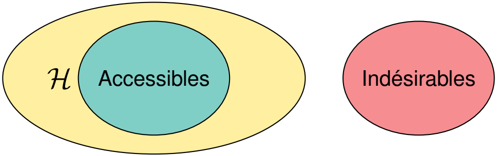



Aucune configuration indésirable accessible

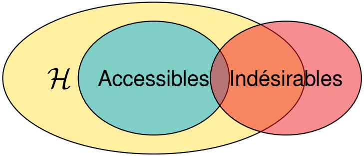



Il existe une configuration indésirable accessible

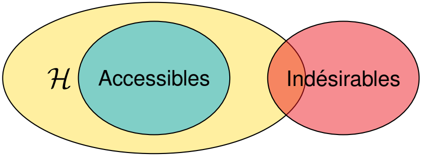
# Sur-approximations




Aucune configuration indésirable accessible




Il existe une configuration indésirable accessible




Aucune configuration indésirable accessible (faux-positif)

# Notre modélisation

- Configuration : un terme

# Notre modélisation

- Configuration : un terme
- Configurations initiales : un langage de termes

# Notre modélisation

- Configuration : un terme
- Configurations initiales : un langage de termes
- Dynamique du système : un système de réécriture

# Notre modélisation

- Configuration : un terme
- Configurations initiales : un langage de termes
- Dynamique du système : un système de réécriture
- Configurations indésirables : un langage de termes

# Sommaire

# Les termes

- Des symboles d'arité fixe : $f^{\backslash 2}$

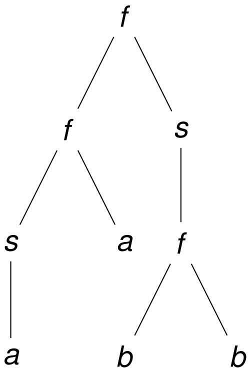
# Les termes

- Des symboles d'arité fixe : $f^{\backslash 2}$
- Un alphabet : $\{a^{\backslash 0}, b^{\backslash 0}, s^{\backslash 1}, h^{\backslash 1}, f^{\backslash 2}\}$

# Les termes

- Des symboles d'arité fixe : $f^{\backslash 2}$
- Un alphabet : $\{a^{\backslash 0}, b^{\backslash 0}, s^{\backslash 1}, h^{\backslash 1}, f^{\backslash 2}\}$
- Un terme : $f(f(s(a), a), s(f(b, b)))$

- Des symboles d'arité fixe : $f^{\backslash 2}$
- Un alphabet : $\{a^{\backslash 0}, b^{\backslash 0}, s^{\backslash 1}, h^{\backslash 1}, f^{\backslash 2}\}$
- Un terme : $f(f(s(a), a), s(f(b, b)))$



# Langages formels

Un ensemble de termes

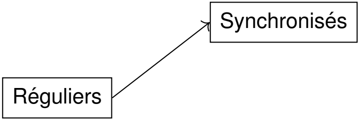
- Langages réguliers : $\{f(s^*(a), s^*(a))\}$

$\boxed{\text{Réguliers}}$

# Langages formels

## Un ensemble de termes

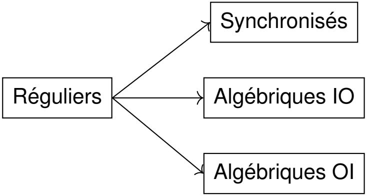
- Langages réguliers : $\{f(s^*(a), s^*(a))\}$
- Langages synchronisés : $\{f(s^n(a), s^n(a)) \mid n \geq 0\}$



# Langages formels

Un ensemble de termes

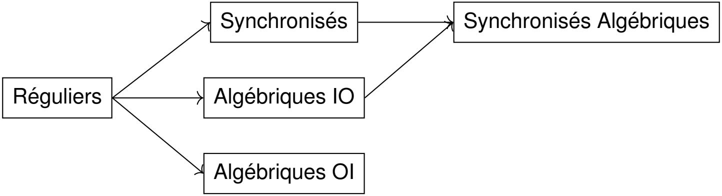
- Langages réguliers : $\{f(s^*(a), s^*(a))\}$
- Langages synchronisés : $\{f(s^n(a), s^n(a)) \mid n \geq 0\}$
- Langages algébriques : $\{s^n(h^n(a)) \mid n \geq 0\}$



# Langages formels

Un ensemble de termes

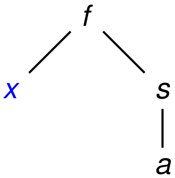
- Langages réguliers : $\{f(s^*(a), s^*(a))\}$
- Langages synchronisés : $\{f(s^n(a), s^n(a)) \mid n \geq 0\}$
- Langages algébriques : $\{s^n(h^n(a)) \mid n \geq 0\}$
- Langages synchronisés algébriques : $\{f(s^n(h^n(a)), s^n(h^n(b))) \mid n \geq 0\}$



# Variables

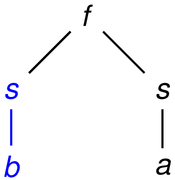
- Ensemble de variables : $\mathcal{X} = \{x, y\}$

# Variables



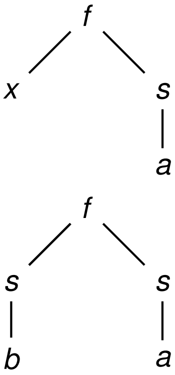

- Ensemble de variables : $\mathcal{X} = \{x, y\}$
- La substitution : $\sigma = (x/s(b))$

# Variables



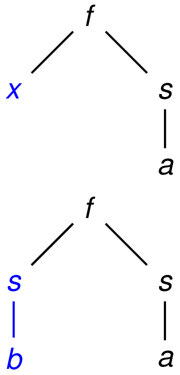

- Ensemble de variables : $\mathcal{X} = \{x, y\}$
- La substitution : $\sigma = (x/s(b))$

# Variables



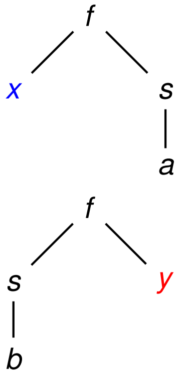

- Ensemble de variables : $\mathcal{X} = \{x, y\}$
- La substitution : $\sigma = (x/s(b))$
- Le filtrage : $\sigma(t) = t'$

# Variables



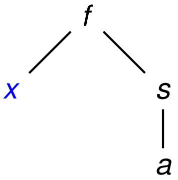

- Ensemble de variables : $\mathcal{X} = \{x, y\}$
- La substitution : $\sigma = (x/s(b))$
- Le filtrage : $\sigma(t) = t'$

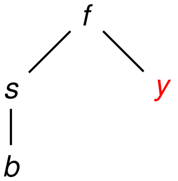
# Variables



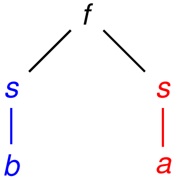

- Ensemble de variables : $\mathcal{X} = \{x, y\}$
- La substitution : $\sigma = (x/s(b))$
- Le filtrage : $\sigma(t) = t'$
- L'unification : $\alpha(t) = \alpha(t')$

- Ensemble de variables : $\mathcal{X} = \{x, y\}$
- La substitution : $\sigma = (x/s(b))$
- Le filtrage : $\sigma(t) = t'$
- L'unification : $\alpha(t) = \alpha(t')$




$$\alpha = (x/s(b), y/s(a))$$



# Systèmes de réécriture

## Règle de réécriture

- $l$ et $r$ des termes

# Systèmes de réécriture

## Règle de réécriture

- $l$ et $r$ des termes
- $l$ n'est pas une variable

# Systèmes de réécriture

## Règle de réécriture

- $l$ et $r$ des termes
- $l$ n'est pas une variable
- $Var(r) \subseteq Var(l)$

# Systèmes de réécriture

## Règle de réécriture

- $l$ et $r$ des termes
- $l$ n'est pas une variable
- $Var(r) \subseteq Var(l)$
- $l \rightarrow r$ est une règle de réécriture

# Systèmes de réécriture

## Règle de réécriture

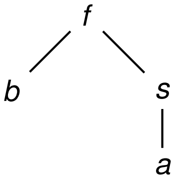
- $l$ et $r$ des termes
- $l$ n'est pas une variable
- $Var(r) \subseteq Var(l)$
- $l \to r$ est une règle de réécriture

$$s(x) \to s(s(x))$$

## Règle de réécriture

- $l$ et $r$ des termes
- $l$ n'est pas une variable
- $Var(r) \subseteq Var(l)$
- $l \to r$ est une règle de réécriture

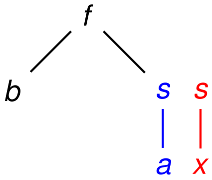
$$s(x) \to s(s(x))$$



## Règle de réécriture

- $l$ et $r$ des termes
- $l$ n'est pas une variable
- $Var(r) \subseteq Var(l)$
- $l \rightarrow r$ est une règle de réécriture

$$s(x) \rightarrow s(s(x))$$
$$\sigma = (x/a)$$

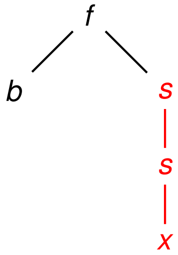


# Systèmes de réécriture

## Règle de réécriture

- $l$ et $r$ des termes
- $l$ n'est pas une variable
- $Var(r) \subseteq Var(l)$
- $l \rightarrow r$ est une règle de réécriture

$$s(x) \rightarrow s(s(x))$$
$$\sigma = (x/a)$$



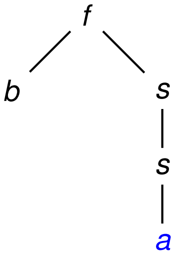
# Systèmes de réécriture

## Règle de réécriture

- $l$ et $r$ des termes
- $l$ n'est pas une variable
- $Var(r) \subseteq Var(l)$
- $l \rightarrow r$ est une règle de réécriture

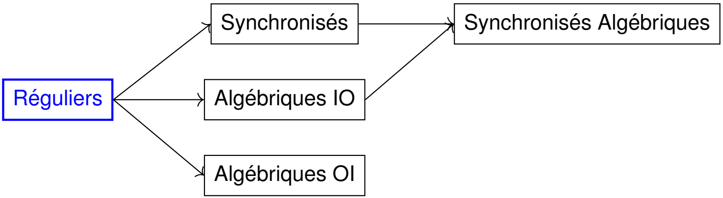
$$s(x) \rightarrow s(s(x))$$

$$\sigma = (x/a)$$



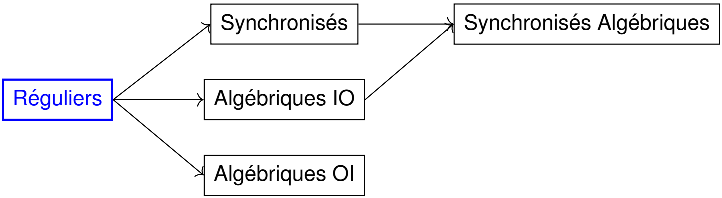
# Sommaire

# Les différentes techniques



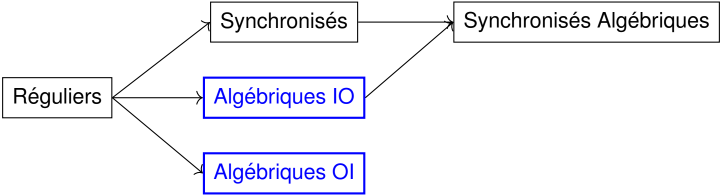

- T. Genet, Decidable approximations of sets of descendants and sets of normal forms, RTA, 1998
- T. Genet et F. Klay, Rewriting for cryptographic protocol verification, CADE, 2000
- T. Genet et V. Rusu, Equational approximations for tree automata completion, JSC, 2010

# Les différentes techniques



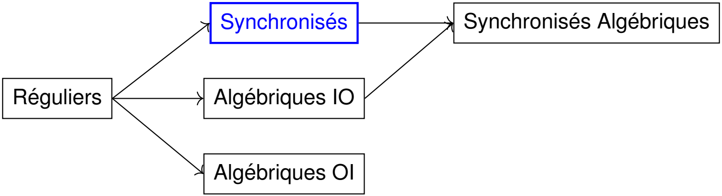

- Y. Boichut et P.-C. Héam, A theoretical limit for safety verification techniques with regular fix-point computations, IPL, 2008

# Les différentes techniques



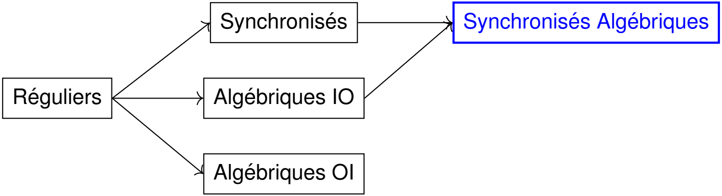

- J. Kochems et L. Ong, Improved functional flow and reachability analyses using indexed linear tree grammars, RTA, 2011

# Les différentes techniques



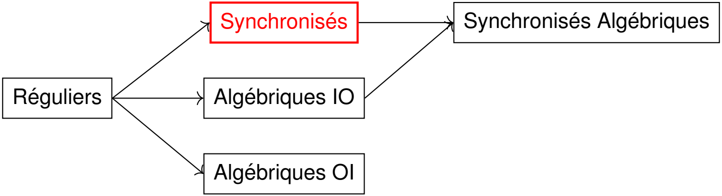

- Y. Boichut, J. Chabin, et P. Réty, Over-approximating descendants by synchronized tree languages, RTA, 2013

# Les différentes techniques




- Y. Boichut, J. Chabin, et P. Réty, Towards more precise rewriting approximations, LATA, 2015

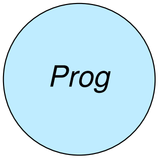
# Les différentes techniques



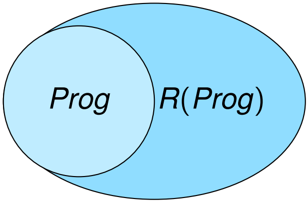

- Y. Boichut, V. Pelletier, et P. Réty, Synchronized tree languages for reachability in non-right-linear term rewrite systems, WRLA, 2016

# Sommaire

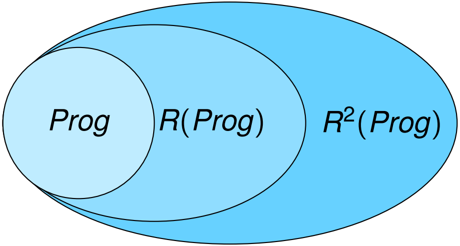
# Méthode de complétion



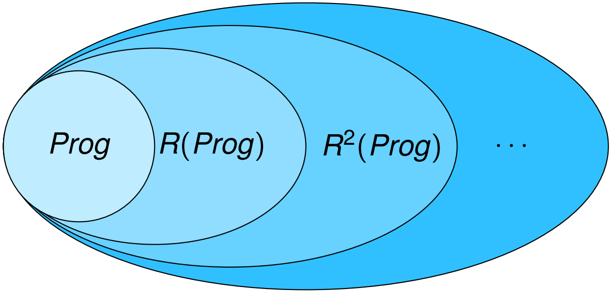
# Méthode de complétion



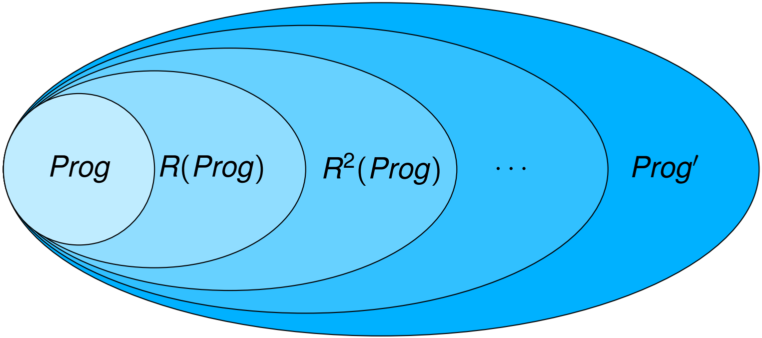
# Méthode de complétion



# Méthode de complétion



# Méthode de complétion



# CS-clause

- Prédicats avec un nombre fixe d'arguments

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$
- Propriétés

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$
- Propriétés
  - plat : tous les arguments des atomes de $B$ sont des variables

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$
- Propriétés
  - plat : tous les arguments des atomes de $B$ sont des variables
  - linéaire : chaque variable dans $B$ n'apparaît qu'une seule fois

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$
- Propriétés
  - plat : tous les arguments des atomes de $B$ sont des variables
  - linéaire : chaque variable dans $B$ n'apparaît qu'une seule fois
  - $\emptyset$ : conjonction d'atomes vide : plat et linéaire

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$
- Propriétés
  - plat : tous les arguments des atomes de $B$ sont des variables
  - linéaire : chaque variable dans $B$ n'apparaît qu'une seule fois
  - $\emptyset$ : conjonction d'atomes vide : plat et linéaire

**CS-clause**

$H \leftarrow B$ avec $B$ plat et linéaire

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$
- Propriétés
  - plat : tous les arguments des atomes de $B$ sont des variables
  - linéaire : chaque variable dans $B$ n'apparaît qu'une seule fois
  - $\emptyset$ : conjonction d'atomes vide : plat et linéaire

**CS-clause**

$H \leftarrow B$ avec $B$ plat et linéaire

**Exemple**

- $P(f(g(x, y)), a) \leftarrow Q(x, y)$ : CS-clause

# CS-clause

- Prédicats avec un nombre fixe d'arguments
- Atome : prédicat avec des termes en argument
- Conjonctions d'atomes : $B = A_1, ..., A_n$
- Propriétés
  - plat : tous les arguments des atomes de $B$ sont des variables
  - linéaire : chaque variable dans $B$ n'apparaît qu'une seule fois
  - $\emptyset$ : conjonction d'atomes vide : plat et linéaire

## CS-clause

$H \leftarrow B$ avec $B$ plat et linéaire

## Exemple

- $P(f(g(x, y)), a) \leftarrow Q(x, y)$ : CS-clause
- $P(x) \leftarrow Q(x), Q_2(g(x))$ : pas une CS-clause

# CS-clause

## Propriétés des CS-clauses

Une clause $H \leftarrow B$ est

# CS-clause

## Propriétés des CS-clauses

Une clause $H \leftarrow B$ est

- copiante si $H$ n'est pas linéaire

# CS-clause

## Propriétés des CS-clauses

Une clause $H \leftarrow B$ est

- copiante si $H$ n'est pas linéaire
- normalisée si tous les arguments sont au plus de profondeur 1

# CS-clause

## Propriétés des CS-clauses

Une clause $H \leftarrow B$ est

- copiante si $H$ n'est pas linéaire
- normalisée si tous les arguments sont au plus de profondeur 1

## Exemple

- $P(f(g(x, a)), x) \leftarrow Q(x)$ : copiante et non normalisée

# CS-clause

## Propriétés des CS-clauses

Une clause $H \leftarrow B$ est

- copiante si $H$ n'est pas linéaire
- normalisée si tous les arguments sont au plus de profondeur 1

## Exemple

- $P(f(g(x, a)), x) \leftarrow Q(x)$ : copiante et non normalisée
- $P(f(x, y), a) \leftarrow Q'(x, y)$ : non copiante et normalisée

# CS-programme

- CS-programme : ensemble de CS-clauses

# CS-programme

- CS-programme : ensemble de CS-clauses
- Opérateurs

# CS-programme

- CS-programme : ensemble de CS-clauses
- Opérateurs

- $G \leadsto G'$ : (dérivation classique)
  $\exists H \leftarrow B \in \mathit{Prog}$ et $\exists \sigma$
  tel que $A \in G$ avec $\sigma(A) = \sigma(H)$ (unification)
  et $G' = \sigma(G)$ avec $\sigma(A)$ remplacé par $\sigma(B)$

- CS-programme : ensemble de CS-clauses
- Opérateurs

- $G \rightsquigarrow G'$ : (dérivation classique)
  $\exists H \leftarrow B \in \mathit{Prog}$ et $\exists \sigma$
  tel que $A \in G$ avec $\sigma(A) = \sigma(H)$ (unification)
  et $G' = \sigma(G)$ avec $\sigma(A)$ remplacé par $\sigma(B)$
- $G \rightarrow G'$ : (dérivation faible)
  comme ci-dessus mais avec $A = \sigma(H)$ (filtrage)

# CS-programme

- CS-programme : ensemble de CS-clauses
- Opérateurs

- $G \leadsto G'$ : (dérivation classique)
  $\exists H \leftarrow B \in Prog$ et $\exists \sigma$
  tel que $A \in G$ avec $\sigma(A) = \sigma(H)$ (unification)
  et $G' = \sigma(G)$ avec $\sigma(A)$ remplacé par $\sigma(B)$
- $G \rightarrow G'$ : (dérivation faible)
  comme ci-dessus mais avec $A = \sigma(H)$ (filtrage)

## Langage d'un CS-programme

Soit $\vec{t}$ des termes clos. $\vec{t} \in \mathcal{L}_{Prog}(P)$ si et seulement si $P(\vec{t}) \leadsto^{*}_{Prog} \emptyset$.

# Exemple

- $Prog =$

# Exemple

- *Prog* =
  1. $P(f(x,y)) \leftarrow Q(x,y)$

# Exemple

- $Prog$ =
  1. $P(f(x,y)) \leftarrow Q(x,y)$
  2. $Q(v(x),w(y)) \leftarrow Q(x,y)$

# Exemple

- $Prog =$
  1. $P(f(x,y)) \leftarrow Q(x,y)$
  2. $Q(v(x), w(y)) \leftarrow Q(x,y)$
  3. $Q(a,a)$

# Exemple

- $Prog =$
  1. $P(f(x,y)) \leftarrow Q(x,y)$
  2. $Q(v(x), w(y)) \leftarrow Q(x,y)$
  3. $Q(a,a)$

- $f(v(a), w(a)) \in \mathcal{L}_{Prog}(P)$ ?

# Exemple

- $Prog$ =
  1. $P(f(x,y)) \leftarrow Q(x,y)$
  2. $Q(v(x),w(y)) \leftarrow Q(x,y)$
  3. $Q(a,a)$

- $f(v(a),w(a)) \in \mathcal{L}_{Prog}(P)$ ?
  - $P(f(v(a),w(a))) \rightsquigarrow Q(v(a),w(a))$

# Exemple

- $Prog =$
  1. $P(f(x,y)) \leftarrow Q(x,y)$
  2. $Q(v(x),w(y)) \leftarrow Q(x,y)$
  3. $Q(a,a)$

- $f(v(a),w(a)) \in \mathcal{L}_{Prog}(P)$ ?
  - $P(f(v(a),w(a))) \leadsto Q(v(a),w(a))$
  - $Q(v(a),w(a)) \leadsto Q(a,a)$

# Exemple

- $Prog$ =
  1. $P(f(x,y)) \leftarrow Q(x,y)$
  2. $Q(v(x),w(y)) \leftarrow Q(x,y)$
  3. $Q(a,a)$

- $f(v(a),w(a)) \in \mathcal{L}_{Prog}(P)$ ?
  - $P(f(v(a),w(a))) \rightsquigarrow Q(v(a),w(a))$
  - $Q(v(a),w(a)) \rightsquigarrow Q(a,a)$
  - $Q(a,a) \rightsquigarrow \emptyset$

# Exemple

- $Prog$ =
  1. $P(f(x, y)) \leftarrow Q(x, y)$
  2. $Q(v(x), w(y)) \leftarrow Q(x, y)$
  3. $Q(a, a)$

- $f(v(a), w(a)) \in \mathcal{L}_{Prog}(P)$ ?
  - $P(f(v(a), w(a))) \rightsquigarrow Q(v(a), w(a))$
  - $Q(v(a), w(a)) \rightsquigarrow Q(a, a)$
  - $Q(a, a) \rightsquigarrow \emptyset$
  - Oui, $f(v(a), w(a)) \in \mathcal{L}_{Prog}(P)$

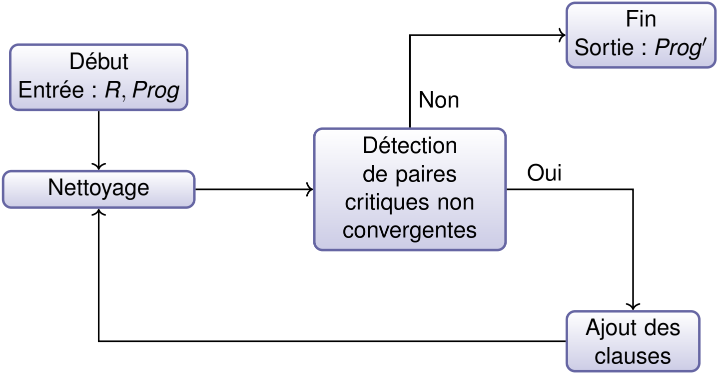
# Exemple

- $Prog$ =
  1. $P(f(x,y)) \leftarrow Q(x,y)$
  2. $Q(v(x),w(y)) \leftarrow Q(x,y)$
  3. $Q(a,a)$

- $f(v(a),w(a)) \in \mathcal{L}_{Prog}(P)$ ?
  - $P(f(v(a),w(a))) \leadsto Q(v(a),w(a))$
  - $Q(v(a),w(a)) \leadsto Q(a,a)$
  - $Q(a,a) \leadsto \emptyset$
  - Oui, $f(v(a),w(a)) \in \mathcal{L}_{Prog}(P)$

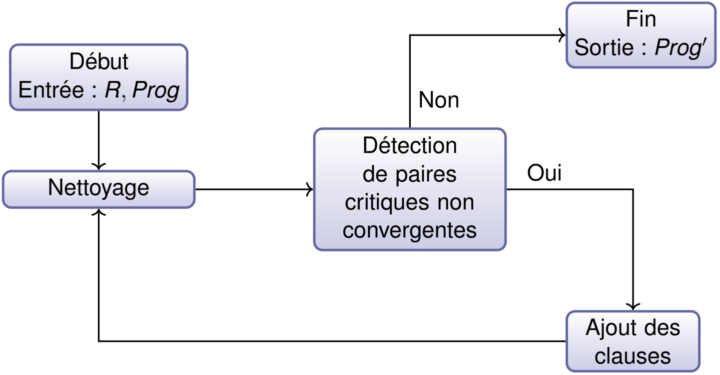
- $\mathcal{L}_{Prog}(P) = \{f(v^n(a),w^n(a)) | n \in \mathbb{N}\}$

# L’algorithme de complétion



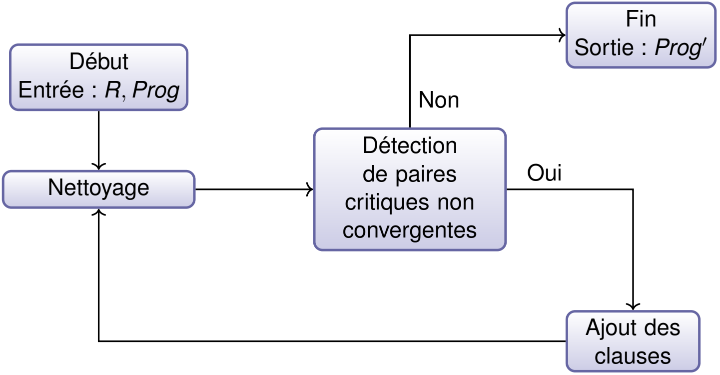

- Un CS-programme initial

# L'algorithme de complétion



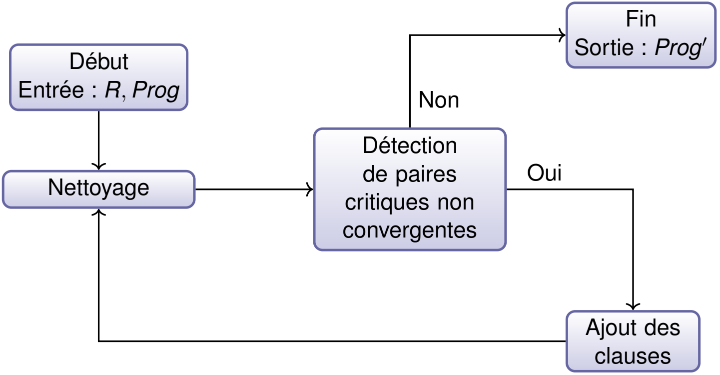

- Un CS-programme initial
  - non copiant

# L'algorithme de complétion



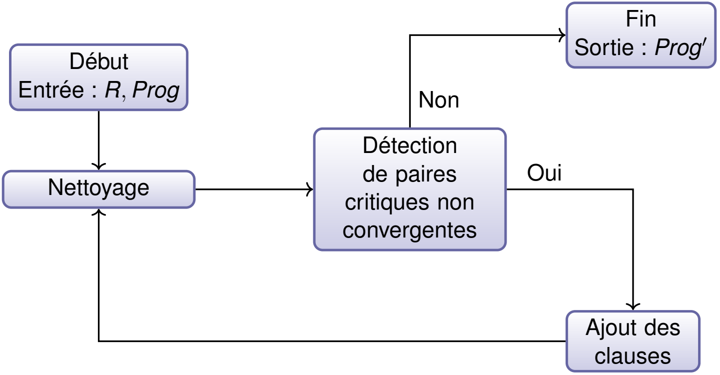

- Un CS-programme initial
  - non copiant
  - normalisé

# L'algorithme de complétion




- Un CS-programme initial
  - non copiant
  - normalisé
- Un système de réécriture

# L'algorithme de complétion




- Un CS-programme initial
  - non copiant
  - normalisé
- Un système de réécriture
  - linéaire

# Paires critiques

- $l \rightarrow r \in R$

# Paires critiques

- $l \rightarrow r \in R$
- $P(..., t_i, ...) \leftarrow B \in Prog$

# Paires critiques

- $l \rightarrow r \in R$
- $P(..., t_i, ...) \leftarrow B \in Prog$

$P(..., l, ...)$

# Paires critiques

- $l \rightarrow r \in R$
- $P(..., t_i, ...) \leftarrow B \in Prog$

$P(..., l, ...) \rightsquigarrow^{+}_{[\sigma]} G$ avec $G$ plat

# Paires critiques

- $l \rightarrow r \in R$
- $P(..., t_i, ...) \leftarrow B \in Prog$

$P(..., l, ...) \rightsquigarrow^{+}_{[\sigma]}$ $G$ avec $G$ plat

$\Big\downarrow$

$\sigma(P(..., r, ...)) \leftarrow G$ : paire critique

- $l \to r \in R$
- $P(..., t_i, ...) \leftarrow B \in Prog$

$P(..., l, ...) \rightsquigarrow^{+}_{[\sigma]}$ $G$ avec $G$ plat

Convergent ?

$\sigma(P(..., r, ...)) \leftarrow G$ : paire critique

# Exemple

- $R : \{g(h(y)) \rightarrow f(y)\}$

# Exemple

- $R : \{g(h(y)) \to f(y)\}$
- $Prog : \{P(g(x)) \leftarrow Q(x).\ \ Q(h(x)) \leftarrow Q'(x).\ \ Q'(a).\}$

# Exemple

- $R : \{g(h(y)) \rightarrow f(y)\}$
- $Prog : \{P(g(x)) \leftarrow Q(x).\ \ Q(h(x)) \leftarrow Q'(x).\ \ Q'(a).\}$

$P(g(h(y)))$

# Exemple

- $R : \{g(h(y)) \to f(y)\}$
- $Prog : \{P(g(x)) \leftarrow Q(x).\ \ Q(h(x)) \leftarrow Q'(x).\ \ Q'(a).\}$

$$P(g(h(y))) \rightsquigarrow Q(h(y))$$

# Exemple

- $R : \{g(h(y)) \rightarrow f(y)\}$
- $Prog : \{P(g(x)) \leftarrow Q(x).\ \ Q(h(x)) \leftarrow Q'(x).\ \ Q'(a).\}$

$$P(g(h(y))) \rightsquigarrow Q(h(y)) \rightsquigarrow Q'(y)$$

# Exemple

- $R : \{g(h(y)) \to f(y)\}$
- $Prog : \{P(g(x)) \leftarrow Q(x).\ \ Q(h(x)) \leftarrow Q'(x).\ \ Q'(a).\}$

$P(g(h(y))) \quad \overset{+}{\underset{[\sigma]}{\rightsquigarrow}} \quad Q'(y)$ et $Q'(y)$ plat

# Exemple

- $R : \{g(h(y)) \to f(y)\}$
- $Prog : \{P(g(x)) \leftarrow Q(x).\ \ Q(h(x)) \leftarrow Q'(x).\ \ Q'(a).\}$

$P(g(h(y))) \overset{+}{\underset{[\sigma]}{\rightsquigarrow}} Q'(y)$ et $Q'(y)$ plat

$\Big\downarrow$

$\sigma(P(f(y))) \leftarrow Q'(y)$ : paire critique

- $R : \{g(h(y)) \rightarrow f(y)\}$
- $Prog : \{P(g(x)) \leftarrow Q(x).\ \ Q(h(x)) \leftarrow Q'(x).\ \ Q'(a).\}$

$P(g(h(y))) \ \overset{+}{\underset{[\sigma]}{\rightsquigarrow}}\ Q'(y)$ et $Q'(y)$ plat

Convergent ?

$\sigma(P(f(y))) \leftarrow Q'(y)$ : paire critique

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x,x)) \leftarrow Q(x)$

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x,x)) \leftarrow Q(x)$ et $Q(b)$

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x,x)) \leftarrow Q(x)$ et $Q(b)$
- $Q(b)$ est convergente

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x, x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x, x)) \leftarrow Q(x)$ et $Q(b)$
- $Q(b)$ est convergente
- On ajoute $P(f(x, x)) \leftarrow Q(x)$ à $Prog$

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x,x)) \leftarrow Q(x)$ et $Q(b)$
- $Q(b)$ est convergente
- On ajoute $P(f(x,x)) \leftarrow Q(x)$ à $Prog$
- Toutes les paires critiques sont convergentes

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x,x)) \leftarrow Q(x)$ et $Q(b)$
- $Q(b)$ est convergente
- On ajoute $P(f(x,x)) \leftarrow Q(x)$ à $Prog$
- Toutes les paires critiques sont convergentes

- $f(a,a) \in \mathcal{L}_{Prog}(P)$

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x,x)) \leftarrow Q(x)$ et $Q(b)$
- $Q(b)$ est convergente
- On ajoute $P(f(x,x)) \leftarrow Q(x)$ à *Prog*
- Toutes les paires critiques sont convergentes

- $f(a,a) \in \mathcal{L}_{Prog}(P)$
- $f(a,a) \rightarrow_R f(b,a)$

# Le problème

- $Prog = \{P(g(x)) \leftarrow Q(x).\ \ Q(a).\ \ Q(b).\}$
- $R = \{g(x) \rightarrow f(x,x),\ \ a \rightarrow b\}$

- 2 paires critiques : $P(f(x,x)) \leftarrow Q(x)$ et $Q(b)$
- $Q(b)$ est convergente
- On ajoute $P(f(x,x)) \leftarrow Q(x)$ à $Prog$
- Toutes les paires critiques sont convergentes

- $f(a,a) \in \mathcal{L}_{Prog}(P)$
- $f(a,a) \rightarrow_R f(b,a)$
- mais $f(b,a)$ et $f(a,b) \notin \mathcal{L}_{Prog}(P)$

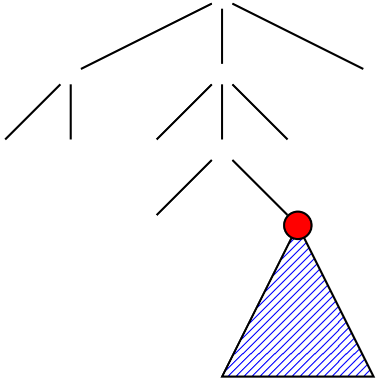
# Sommaire

# Stratégie innermost

- Restriction des réécritures

# Stratégie innermost

- Restriction des réécritures
- Réécritures de sous-termes non réductibles



# Irréducibilité forte

## Irréducibilité forte

Pour tout $l \to r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.

# Irréducibilité forte

## Irréducibilité forte

Pour tout $l \to r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

# Irréducibilité forte

## Irréducibilité forte

Pour tout $l \to r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

## Exemple

- $f(b) \to b$

## Irréductibilité forte

Pour tout $l \to r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

## Exemple

- $f(b) \to b$
- $t = f(x)$ :

# Irréducibilité forte

## Irréducibilité forte

Pour tout $l \rightarrow r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

## Exemple

- $f(b) \rightarrow b$
- $t = f(x)$ : irréducible

# Irréductibilité forte

## Irréductibilité forte

Pour tout $l \to r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

## Exemple

- $f(b) \to b$
- $t = f(x)$ : irréductible
- $\theta = (x/a)$ :

# Irréducibilité forte

## Irréducibilité forte

Pour tout $l \rightarrow r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

## Exemple

- $f(b) \rightarrow b$
- $t = f(x)$ : irréductible
- $\theta = (x/a)$ : fortement irréductible

## Irréductibilité forte

Pour tout $l \to r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

## Exemple

- $f(b) \to b$
- $t = f(x)$ : irréductible
- $\theta = (x/a)$ : fortement irréductible
- $\theta(t) = f(a)$ :

# Irréductibilité forte

## Irréductibilité forte

Pour tout $l \to r \in R$.

- Un terme $t$ est *fortement irréductible* (par $R$) si aucun de ses sous-termes ne peut s'unifier avec $l$.
- Une substitution $\theta$ est *fortement irréductible* si pour tout $x \in \mathcal{X}$, $\theta(x)$ est fortement irréductible.

## Exemple

- $f(b) \to b$
- $t = f(x)$ : irréductible
- $\theta = (x/a)$ : fortement irréductible
- $\theta(t) = f(a)$ : fortement irréductible

## NC et SNC

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \rightsquigarrow_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.

## NC et SNC

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \leadsto_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.
- Une dérivation est NC (resp. SNC) si toutes ses étapes le sont.

**NC et SNC**

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \leadsto_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.
- Une dérivation est NC (resp. SNC) si toutes ses étapes le sont.

Une clause non copiante est SNC

## Dérivations non copiantes (NC) et fortement non copiantes (SNC)

### NC et SNC

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \leadsto_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.
- Une dérivation est NC (resp. SNC) si toutes ses étapes le sont.

Une clause non copiante est SNC

### Exemple

- $P(g(x, x)) \leftarrow Q(x)$

**NC et SNC**

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \rightsquigarrow_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.
- Une dérivation est NC (resp. SNC) si toutes ses étapes le sont.

Une clause non copiante est SNC

**Exemple**

- $P(g(x, x)) \leftarrow Q(x)$
- $R = \{h(a) \rightarrow b\}$

## Dérivations non copiantes (NC) et fortement non copiantes (SNC)

### NC et SNC

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \rightsquigarrow_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.
- Une dérivation est NC (resp. SNC) si toutes ses étapes le sont.

Une clause non copiante est SNC

### Exemple

- $P(g(x, x)) \leftarrow Q(x)$
- $R = \{h(a) \rightarrow b\}$
- $P(g(h(y), h(y)))$

# Dérivations non copiantes (NC) et fortement non copiantes (SNC)

## NC et SNC

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \rightsquigarrow_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.
- Une dérivation est NC (resp. SNC) si toutes ses étapes le sont.

Une clause non copiante est SNC

## Exemple

- $P(g(x,x)) \leftarrow Q(x)$
- $R = \{h(a) \rightarrow b\}$
- $P(g(h(y), h(y))) \rightsquigarrow Q(h(y))$ :

### NC et SNC

Soit $A$ un atome ($A$ peut contenir des variables).

- L'étape $A \leadsto_{[H \leftarrow B, \sigma]} G$ est NC (resp. SNC) si pour toute variable $x \in Var(H)$ non linéaire, $x$ est irréductible (resp. fortement irréductible) par $R$.
- Une dérivation est NC (resp. SNC) si toutes ses étapes le sont.

Une clause non copiante est SNC

### Exemple

- $P(g(x,x)) \leftarrow Q(x)$
- $R = \{h(a) \to b\}$
- $P(g(h(y), h(y))) \leadsto Q(h(y))$ : NC mais pas SNC

# Clôture par réécriture innermost

**Théorème [WRLA, 2016, Y. Boichut, V. Pelletier et P. Réty]**

Soit $R$ un système de réécriture linéaire gauche.

# Clôture par réécriture innermost

**Théorème [WRLA, 2016, Y. Boichut, V. Pelletier et P. Réty]**

Soit $R$ un système de réécriture linéaire gauche.
Soit *Prog* un CS-programme normalisé et non copiant.

# Clôture par réécriture innermost

**Théorème [WRLA, 2016, Y. Boichut, V. Pelletier et P. Réty]**

Soit $R$ un système de réécriture linéaire gauche.
Soit $Prog$ un CS-programme normalisé et non copiant.
Soit $Prog' \supseteq Prog$ un CS-programme

# Clôture par réécriture innermost

**Théorème [WRLA, 2016, Y. Boichut, V. Pelletier et P. Réty]**

Soit $R$ un système de réécriture linéaire gauche.
Soit *Prog* un CS-programme normalisé et non copiant.
Soit $Prog' \supseteq Prog$ un CS-programme
tel que toutes les paires critiques de *Prog* sont convergentes
par dérivations SNC dans $Prog'$.

# Clôture par réécriture innermost

**Théorème [WRLA, 2016, Y. Boichut, V. Pelletier et P. Réty]**

Soit $R$ un système de réécriture linéaire gauche.
Soit *Prog* un CS-programme normalisé et non copiant.
Soit $Prog' \supseteq Prog$ un CS-programme
tel que toutes les paires critiques de *Prog* sont convergentes
par dérivations SNC dans $Prog'$.
Si un terme $t \in \mathcal{L}_{Prog}(P)$ et $t \rightarrow_R^* t'$ avec une stratégie innermost, alors $t' \in \mathcal{L}_{Prog'}(P)$.

# Sommaire

# Origine des clauses copiantes

- Les réécritures non couvertes proviennent des clauses copiantes

# Origine des clauses copiantes

- Les réécritures non couvertes proviennent des clauses copiantes
- Les clauses copiantes proviennent

# Origine des clauses copiantes

- Les réécritures non couvertes proviennent des clauses copiantes
- Les clauses copiantes proviennent
  - de clauses copiantes présentes initialement

# Origine des clauses copiantes

- Les réécritures non couvertes proviennent des clauses copiantes
- Les clauses copiantes proviennent
  - de clauses copiantes présentes initialement
  - du système de réécriture non linéaire droit

# Origine des clauses copiantes

- Les réécritures non couvertes proviennent des clauses copiantes
- Les clauses copiantes proviennent
  - de clauses copiantes présentes initialement
  - du système de réécriture non linéaire droit

**Solution**

# Origine des clauses copiantes

- Les réécritures non couvertes proviennent des clauses copiantes
- Les clauses copiantes proviennent
  - de clauses copiantes présentes initialement
  - du système de réécriture non linéaire droit

**Solution**

- transformer les clauses copiantes en non copiantes (*uncopying*)

# Exemple

- $Prog =$

# Exemple

- *Prog* =
  1. $P(f(x,x)) \leftarrow Q(x)$

# Exemple

- $Prog =$
  1. $P(f(x, x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$

# Exemple

- $Prog =$
  1. $P(f(x,x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$
  3. $Q(a)$

# Exemple

- $Prog =$
  1. $P(f(x,x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$
  3. $Q(a)$

- $uncopying(P(f(x,x)) \leftarrow Q(x))$

# Exemple

- $Prog =$
  1. $P(f(x,x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$
  3. $Q(a)$

- $uncopying(P(f(x,x)) \leftarrow Q(x))$
- $P(f(x,y)) \leftarrow Q^2(x,y)$

# Exemple

- $Prog =$
  1. $P(f(x,x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$
  3. $Q(a)$

- $uncopying(P(f(x,x)) \leftarrow Q(x))$
- $P(f(x,y)) \leftarrow Q^2(x,y)$
- $Q^2(s(x),s(y)) \leftarrow Q^2(x,y)$

# Exemple

- $Prog =$
  1. $P(f(x,x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$
  3. $Q(a)$

- $uncopying(P(f(x,x)) \leftarrow Q(x))$
- $P(f(x,y)) \leftarrow Q^2(x,y)$
- $Q^2(s(x),s(y)) \leftarrow Q^2(x,y)$
- $Q^2(a,a) \leftarrow$

# Exemple

- $Prog =$
  1. $P(f(x,x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$
  3. $Q(a)$

- $uncopying(P(f(x,x)) \leftarrow Q(x))$
- $P(f(x,y)) \leftarrow Q^2(x,y)$
- $Q^2(s(x),s(y)) \leftarrow Q^2(x,y)$
- $Q^2(a,a) \leftarrow$

- $\mathcal{L}_{Prog}(Q^2) = \{t \cdot t' \mid t,t' \in \mathcal{L}_{Prog}(Q) \wedge t = t'\}$

# Exemple

- $Prog =$
  1. $P(f(x,x)) \leftarrow Q(x)$
  2. $Q(s(x)) \leftarrow Q(x)$
  3. $Q(a)$

- $uncopying(P(f(x,x)) \leftarrow Q(x))$
- $P(f(x,y)) \leftarrow Q^2(x,y)$
- $Q^2(s(x),s(y)) \leftarrow Q^2(x,y)$
- $Q^2(a,a) \leftarrow$

- $\mathcal{L}_{Prog}(Q^2) = \{t \cdot t' \mid t, t' \in \mathcal{L}_{Prog}(Q) \wedge t = t'\}$
- Plus de clauses copiantes

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- $Prog =$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- *Prog* =
  1. $P(c(x,x)) \leftarrow P(x)$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- *Prog* =
  1. $P(c(x, x)) \leftarrow P(x)$
  2. $P(a)$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- *Prog* =
  1. $P(c(x, x)) \leftarrow P(x)$
  2. $P(a)$
- $uncopying(P(c(x, x)) \leftarrow P(x))$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- *Prog* =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- $Prog$ =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$
- $P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y)$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- *Prog* =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$
- $P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y)$
- $P^2(a,a) \leftarrow$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- $Prog$ =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$
- $P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y)$
- $P^2(a,a) \leftarrow$

- $uncopying(P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y))$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- $Prog$ =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$
- $P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y)$
- $P^2(a,a) \leftarrow$

- $uncopying(P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y))$
- $P^2(c(x,x'),c(y,y')) \leftarrow P^4(x,x',y,y')$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- *Prog* =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$
- $P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y)$
- $P^2(a,a) \leftarrow$

- $uncopying(P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y))$
- $P^2(c(x,x'),c(y,y')) \leftarrow P^4(x,x',y,y')$
- $P^4(c(x,x),c(x',x'),c(y,y),c(y',y')) \leftarrow P^4(x,x',y,y')$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- $Prog$ =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$
- $P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y)$
- $P^2(a,a) \leftarrow$

- $uncopying(P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y))$
- $P^2(c(x,x'),c(y,y')) \leftarrow P^4(x,x',y,y')$
- $P^4(c(x,x),c(x',x'),c(y,y),c(y',y')) \leftarrow P^4(x,x',y,y')$
- $P^4(a,a,a,a) \leftarrow$

# Terminaison de l'élimination de clauses copiantes

- Ne termine pas dans le cas général
- *Prog* =
  1. $P(c(x,x)) \leftarrow P(x)$
  2. $P(a)$

- $uncopying(P(c(x,x)) \leftarrow P(x))$
- $P(c(x,y)) \leftarrow P^2(x,y)$
- $P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y)$
- $P^2(a,a) \leftarrow$

- $uncopying(P^2(c(x,x),c(y,y)) \leftarrow P^2(x,y))$
- $P^2(c(x,x'),c(y,y')) \leftarrow P^4(x,x',y,y')$
- $P^4(c(x,x),c(x',x'),c(y,y),c(y',y')) \leftarrow P^4(x,x',y,y')$
- $P^4(a,a,a,a) \leftarrow$

- et ainsi de suite

# Forcer la terminaison

- $UncopyingLimit = 2$

# Forcer la terminaison

- $UncopyingLimit = 2$
- $P^2(c(x, x'), c(y, y')) \leftarrow P^2(x, x'), P(y), P(y')$

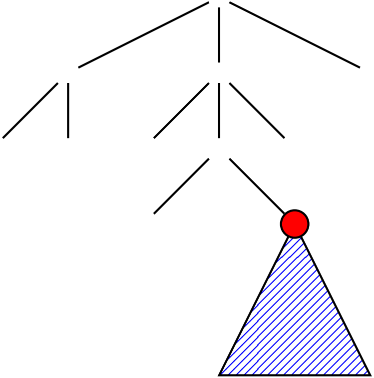
# Forcer la terminaison

- $UncopyingLimit = 2$
- $P^2(c(x, x'), c(y, y')) \leftarrow P^2(x, x'), P(y), P(y')$
- $\{t \cdot t \cdot t' \cdot t'' \mid t, t', t'' \in L(P)\} \supset L(P^4)$

# Sommaire

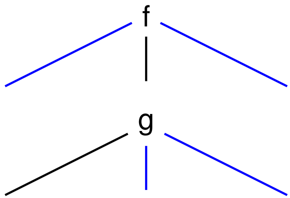
# Stratégies de réécriture

- *Innermost*



# Stratégies de réécriture

- *Innermost*
- *context-sensitive* (CSTRS)




J. Giesl et A. Middeldorp, Transformation techniques for context-sensitive rewrite systems, JFP, 2004

# Stratégies de réécriture

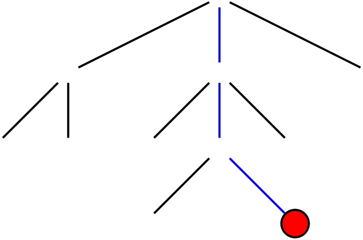
- *Innermost*
- *context-sensitive* (CSTRS)
- stratégie programmable

- Opérations élémentaires
- Opérations de contrôle de combinaison
- Opérations transversales

H. Cirstea, S. Lenglet et P.-E. Moreau, A faithful encoding of programmable strategies into term rewriting systems, RTA, 2015

# Stratégies de réécriture

- *Innermost*
- *context-sensitive* (CSTRS)
- stratégie programmable
- *prefix-constrained* (pCTRS)



N. Andrianarivelo, V. Pelletier et P. Réty, Transforming prefix-constrained or controlled rewrite systems, SCSS, 2017

# pCTRS

- $L : l \to r$

# pCTRS

- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$

# pCTRS

- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

## Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$

# pCTRS

- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

## Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{L : a \to b\}$

- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

**Exemple**

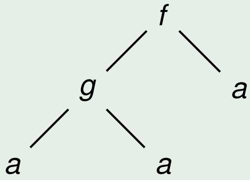
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{L : a \to b\}$
- $L = (\langle f, 1\rangle \cdot \langle g, 2\rangle)^*$

# pCTRS

- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

## Exemple

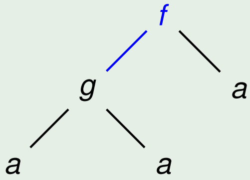
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{L : a \to b\}$
- $L = (\langle f, 1\rangle \cdot \langle g, 2\rangle)^*$



- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

## Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{L : a \to b\}$
- $L = (\langle f, 1\rangle \cdot \langle g, 2\rangle)^*$

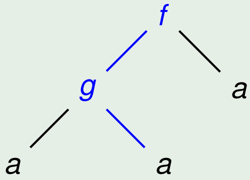


# pCTRS

- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

## Exemple

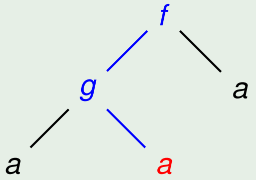
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{L : a \to b\}$
- $L = (\langle f, 1\rangle \cdot \langle g, 2\rangle)^*$



- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

## Exemple

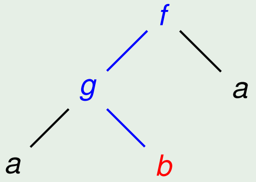
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{L : a \to b\}$
- $L = (\langle f, 1\rangle \cdot \langle g, 2\rangle)^*$



- $L : l \to r$
- $Dir(\Sigma) = \{\langle f, i\rangle \mid f \in \Sigma, 1 \leq i \leq arity(f)\}$
- $L$ automate de mots construit sur $Dir(\Sigma)$

## Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{L : a \to b\}$
- $L = (\langle f, 1\rangle \cdot \langle g, 2\rangle)^*$



# Transformation de pCTRS

- $R' = R'_1 \cup R'_2 \cup R'_3 \cup R'_4$
- $R'_1$ : Règles d'amorçage
- $R'_2$ : Règles de parcours de l'automate
- $R'_3$ : Règles de réécriture du pCTRS
- $R'_4$ : Règles de remontée du jeton

# Transformation de pCTRS

- $R' = R'_1 \cup R'_2 \cup R'_3 \cup R'_4$
- $R'_1 = \{top(j(x)) \to top(q_I(x)) \mid q_I \in Q_I\}$
- $R'_2$ : Règles de parcours de l'automate
- $R'_3$ : Règles de réécriture du pCTRS
- $R'_4$ : Règles de remontée du jeton

# Transformation de pCTRS

- $R' = R'_1 \cup R'_2 \cup R'_3 \cup R'_4$
- $R'_1 = \{top(j(x)) \to top(q_I(x)) \mid q_I \in Q_I\}$
- $R'_2 = \{q(f(x_1, \ldots, x_n)) \to \delta(x_1, \ldots, q'(x_i), \ldots, x_n) \mid$
  $\delta = (q, \langle f, i \rangle, q') \in \Delta\}$
- $R'_3$ : Règles de réécriture du pCTRS
- $R'_4$ : Règles de remontée du jeton

# Transformation de pCTRS

- $R' = R'_1 \cup R'_2 \cup R'_3 \cup R'_4$
- $R'_1 = \{top(j(x)) \rightarrow top(q_I(x)) \mid q_I \in Q_I\}$
- $R'_2 = \{q(f(x_1, \ldots, x_n)) \rightarrow \delta(x_1, \ldots, q'(x_i), \ldots, x_n) \mid$
  $\delta = (q, \langle f, i \rangle, q') \in \Delta\}$
- $R'_3 = \{q_f(l_k) \rightarrow j(r_k) \mid q_f \in Q_f^k, (L_k : l_k \rightarrow r_k) \in R\}$
- $R'_4$ : Règles de remontée du jeton

# Transformation de pCTRS

- $R' = R'_1 \cup R'_2 \cup R'_3 \cup R'_4$
- $R'_1 = \{top(j(x)) \to top(q_I(x)) \mid q_I \in Q_I\}$
- $R'_2 = \{q(f(x_1, \ldots, x_n)) \to \delta(x_1, \ldots, q'(x_i), \ldots, x_n) \mid$
  $\delta = (q, \langle f, i \rangle, q') \in \Delta\}$
- $R'_3 = \{q_f(l_k) \to j(r_k) \mid q_f \in Q_f^k, (L_k : l_k \to r_k) \in R\}$
- $R'_4 = \{\delta(x_1, \ldots, j(x_i), \ldots, x_n) \to j(f(x_1, \ldots, x_i, \ldots, x_n)) \mid$
  $\delta = (q, \langle f, i \rangle, q') \in \Delta\}$

- $R' = R'_1 \cup R'_2 \cup R'_3 \cup R'_4$
- $R'_1 = \{top(j(x)) \to top(q_I(x)) \mid q_I \in Q_I\}$
- $R'_2 = \{q(f(x_1, \ldots, x_n)) \to \delta(x_1, \ldots, q'(x_i), \ldots, x_n) \mid$ $\delta = (q, \langle f, i \rangle, q') \in \Delta\}$
- $R'_3 = \{q_f(l_k) \to j(r_k) \mid q_f \in Q_f^k, (L_k : l_k \to r_k) \in R\}$
- $R'_4 = \{\delta(x_1, \ldots, j(x_i), \ldots, x_n) \to j(f(x_1, \ldots, x_i, \ldots, x_n)) \mid$ $\delta = (q, \langle f, i \rangle, q') \in \Delta\}$

**Théorème : [SCSS, 2017, N. Andrianarivelo, V. Pelletier et P. Réty]**

Soit $t \in T(\Sigma)$. On a $t \to^*_{R_{pc}} t'$ si et seulement si $top(j(t)) \to^*_{R'} top(j(t'))$.

# Transformation de pCTRS

- $R' = R'_1 \cup R'_2 \cup R'_3 \cup R'_4$
- $R'_1 = \{top(j(x)) \to top(q_l(x)) \mid q_l \in Q_l\}$
- $R'_2 = \{q(f(x_1, \ldots, x_n)) \to \delta(x_1, \ldots, q'(x_i), \ldots, x_n) \mid$
  $\delta = (q, \langle f, i\rangle, q') \in \Delta\}$
- $R'_3 = \{q_f(l_k) \to j(r_k) \mid q_f \in Q_f^k, \, (L_k : l_k \to r_k) \in R\}$
- $R'_4 = \{\delta(x_1, \ldots, j(x_i), \ldots, x_n) \to j(f(x_1, \ldots, x_i, \ldots, x_n)) \mid$
  $\delta = (q, \langle f, i\rangle, q') \in \Delta\}$

**Théorème : [SCSS, 2017, N. Andrianarivelo, V. Pelletier et P. Réty]**

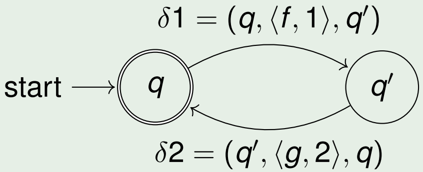
Soit $t \in T(\Sigma)$. On a $t \to^*_{R_{pc}} t'$ si et seulement si $top(j(t)) \to^*_{R'} top(j(t'))$.

**Théorème : [SCSS, 2017, N. Andrianarivelo, V. Pelletier et P. Réty]**

Le pCTRS $R$ est terminant sur $\Sigma$ si et seulement si $R'$ est terminant sur $\Sigma'$.

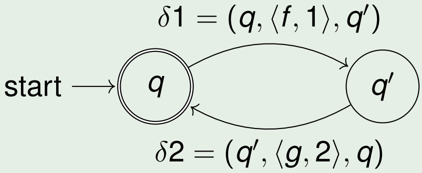
# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$



# Exemple

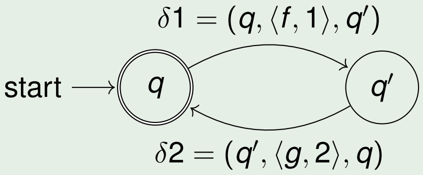
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f, 1\rangle.\langle g, 2\rangle)^* : a \rightarrow b\}$
- $Dir(\Sigma) = \{\langle f, 1\rangle, \langle f, 2\rangle, \langle g, 1\rangle, \langle g, 2\rangle\}$




- $R'_1 = \{top(j(x)) \rightarrow top(q(x))\}$

# Exemple

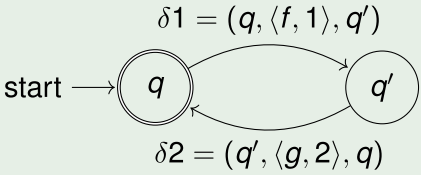
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$




- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x), y), q'(g(x,y)) \to \delta_2(x, q(y))\}$

# Exemple

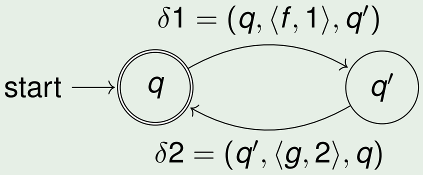
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f, 1\rangle . \langle g, 2\rangle)^* : a \to b\}$
- $Dir(\Sigma) =$
  $\{\langle f, 1\rangle, \langle f, 2\rangle, \langle g, 1\rangle, \langle g, 2\rangle\}$




- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x, y)) \to \delta_1(q'(x), y),$
  $q'(g(x, y)) \to \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$

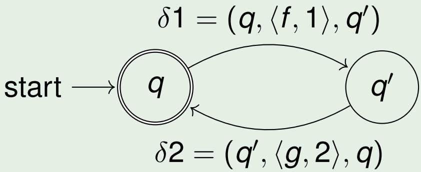
# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

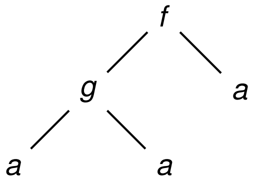



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x),y),$
  $q'(g(x,y)) \to \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x),y) \to j(f(x,y)),$
  $\delta_2(x, j(y)) \to j(g(x,y))\}$

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f, 1\rangle.\langle g, 2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f, 1\rangle, \langle f, 2\rangle, \langle g, 1\rangle, \langle g, 2\rangle\}$

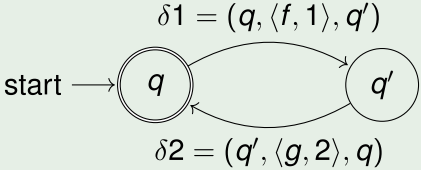



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x, y)) \to \delta_1(q'(x), y), q'(g(x, y)) \to \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x), y) \to j(f(x, y)), \delta_2(x, j(y)) \to j(g(x, y))\}$

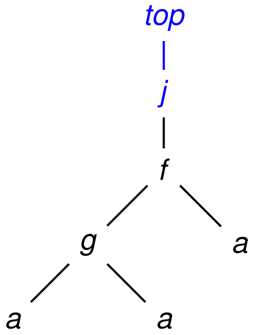


# Exemple

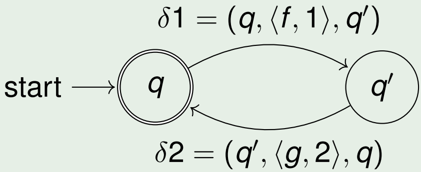
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f, 1\rangle.\langle g, 2\rangle)^* : a \rightarrow b\}$
- $Dir(\Sigma) = \{\langle f, 1\rangle, \langle f, 2\rangle, \langle g, 1\rangle, \langle g, 2\rangle\}$



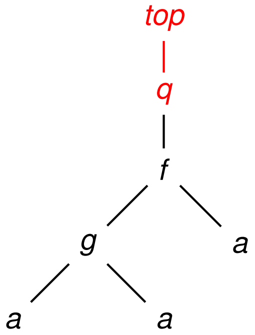

- $R'_1 = \{top(j(x)) \rightarrow top(q(x))\}$
- $R'_2 = \{q(f(x, y)) \rightarrow \delta_1(q'(x), y), q'(g(x, y)) \rightarrow \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \rightarrow j(b)\}$
- $R'_4 = \{\delta_1(j(x), y) \rightarrow j(f(x, y)), \delta_2(x, j(y)) \rightarrow j(g(x, y))\}$



- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

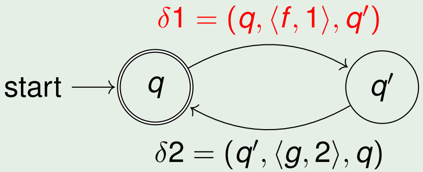



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x),y), q'(g(x,y)) \to \delta_2(x,q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x),y) \to j(f(x,y)), \delta_2(x,j(y)) \to j(g(x,y))\}$

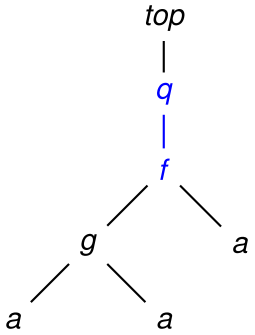


# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \rightarrow b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

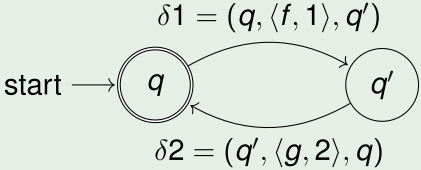



- $R'_1 = \{top(j(x)) \rightarrow top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \rightarrow \delta_1(q'(x),y), q'(g(x,y)) \rightarrow \delta_2(x,q(y))\}$
- $R'_3 = \{q(a) \rightarrow j(b)\}$
- $R'_4 = \{\delta_1(j(x),y) \rightarrow j(f(x,y)), \delta_2(x,j(y)) \rightarrow j(g(x,y))\}$

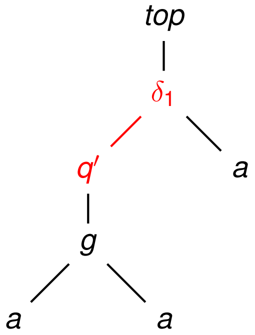


# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \rightarrow b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

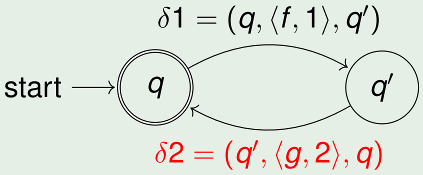



- $R'_1 = \{top(j(x)) \rightarrow top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \rightarrow \delta_1(q'(x),y), q'(g(x,y)) \rightarrow \delta_2(x,q(y))\}$
- $R'_3 = \{q(a) \rightarrow j(b)\}$
- $R'_4 = \{\delta_1(j(x),y) \rightarrow j(f(x,y)), \delta_2(x,j(y)) \rightarrow j(g(x,y))\}$

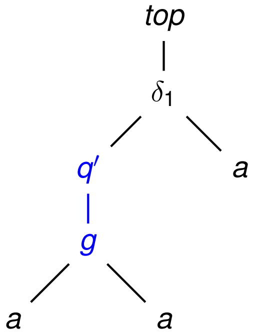


# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

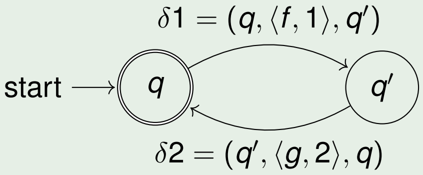



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x), y),$ $q'(g(x,y)) \to \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x), y) \to j(f(x,y)),$ $\delta_2(x, j(y)) \to j(g(x,y))\}$

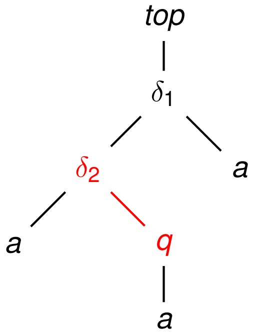


# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f, 1\rangle.\langle g, 2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f, 1\rangle, \langle f, 2\rangle, \langle g, 1\rangle, \langle g, 2\rangle\}$

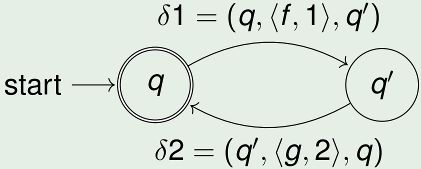



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x, y)) \to \delta_1(q'(x), y), q'(g(x, y)) \to \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x), y) \to j(f(x, y)), \delta_2(x, j(y)) \to j(g(x, y))\}$

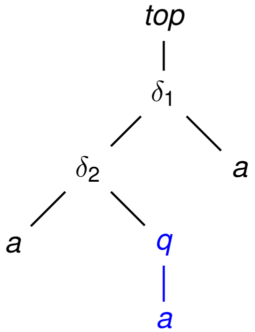


# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

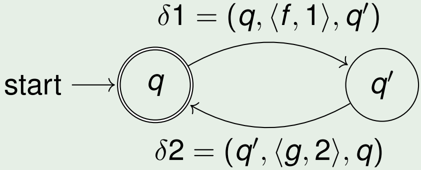



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x),y), q'(g(x,y)) \to \delta_2(x,q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x),y) \to j(f(x,y)), \delta_2(x,j(y)) \to j(g(x,y))\}$

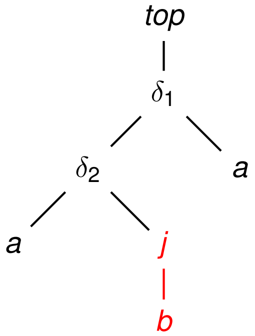


# Exemple

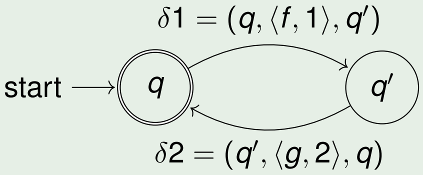
- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f, 1\rangle.\langle g, 2\rangle)^* : a \rightarrow b\}$
- $Dir(\Sigma) = \{\langle f, 1\rangle, \langle f, 2\rangle, \langle g, 1\rangle, \langle g, 2\rangle\}$



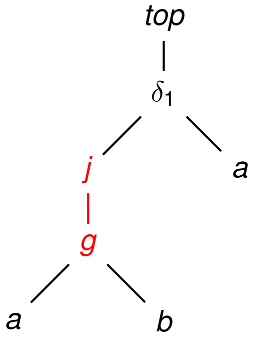

- $R'_1 = \{top(j(x)) \rightarrow top(q(x))\}$
- $R'_2 = \{q(f(x, y)) \rightarrow \delta_1(q'(x), y), q'(g(x, y)) \rightarrow \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \rightarrow j(b)\}$
- $R'_4 = \{\delta_1(j(x), y) \rightarrow j(f(x, y)), \delta_2(x, j(y)) \rightarrow j(g(x, y))\}$



- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

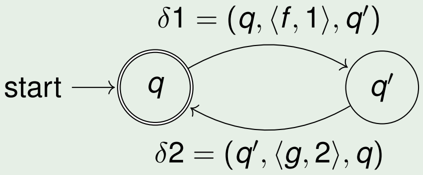



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x), y), q'(g(x,y)) \to \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x), y) \to j(f(x,y)), \delta_2(x, j(y)) \to j(g(x,y))\}$

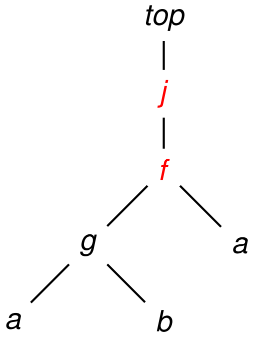


# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$

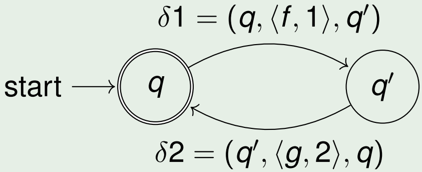



- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x),y),$ $q'(g(x,y)) \to \delta_2(x, q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x),y) \to j(f(x,y)),$ $\delta_2(x, j(y)) \to j(g(x,y))\}$

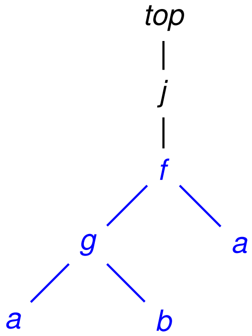


# Exemple

- $\Sigma = \{f^{\backslash 2}, g^{\backslash 2}, a^{\backslash 0}, b^{\backslash 0}\}$
- $R = \{(\langle f,1\rangle.\langle g,2\rangle)^* : a \to b\}$
- $Dir(\Sigma) = \{\langle f,1\rangle, \langle f,2\rangle, \langle g,1\rangle, \langle g,2\rangle\}$




- $R'_1 = \{top(j(x)) \to top(q(x))\}$
- $R'_2 = \{q(f(x,y)) \to \delta_1(q'(x),y), q'(g(x,y)) \to \delta_2(x,q(y))\}$
- $R'_3 = \{q(a) \to j(b)\}$
- $R'_4 = \{\delta_1(j(x),y) \to j(f(x,y)), \delta_2(x,j(y)) \to j(g(x,y))\}$



# Sommaire

# Conclusion

- Calcul de sur-approximations non régulières

# Conclusion

- Calcul de sur-approximations non régulières
    - Descendants innermost

# Conclusion

- Calcul de sur-approximations non régulières
  - Descendants innermost
  - Élimination de clauses copiantes

# Conclusion

- Calcul de sur-approximations non régulières
  - Descendants innermost
  - Élimination de clauses copiantes
- Transformation de systèmes de réécriture avec stratégie

# Conclusion

- Calcul de sur-approximations non régulières
  - Descendants innermost
  - Élimination de clauses copiantes
- Transformation de systèmes de réécriture avec stratégie
  - Transformation de pCTRS

# Conclusion

- Calcul de sur-approximations non régulières
  - Descendants innermost
  - Élimination de clauses copiantes
- Transformation de systèmes de réécriture avec stratégie
  - Transformation de pCTRS
  - Amélioration de la transformation des csTRS

# Conclusion

- Calcul de sur-approximations non régulières
  - Descendants innermost
  - Élimination de clauses copiantes
- Transformation de systèmes de réécriture avec stratégie
  - Transformation de pCTRS
  - Amélioration de la transformation des csTRS
- Transformation d'un programme logique en CS-programme avec sur-approximation

# Perspectives

- Calcul de sur-approximations non régulières

# Perspectives

- Calcul de sur-approximations non régulières
  - Améliorer la précision des approximations

# Perspectives

- Calcul de sur-approximations non régulières
  - Améliorer la précision des approximations
  - Développer des heuristiques

# Perspectives

- Calcul de sur-approximations non régulières
  - Améliorer la précision des approximations
  - Développer des heuristiques
  - Porter les résultats existants sur les réguliers

# Perspectives

- Calcul de sur-approximations non régulières
  - Améliorer la précision des approximations
  - Développer des heuristiques
  - Porter les résultats existants sur les réguliers
- Transformation de pCTRS

# Perspectives

- Calcul de sur-approximations non régulières
  - Améliorer la précision des approximations
  - Développer des heuristiques
  - Porter les résultats existants sur les réguliers
- Transformation de pCTRS
  - Explorer d'autres stratégies

# Perspectives

- Calcul de sur-approximations non régulières
  - Améliorer la précision des approximations
  - Développer des heuristiques
  - Porter les résultats existants sur les réguliers
- Transformation de pCTRS
  - Explorer d'autres stratégies
- Transformation d'un programme logique en CS-programme avec sur-approximation

# Perspectives

- Calcul de sur-approximations non régulières
  - Améliorer la précision des approximations
  - Développer des heuristiques
  - Porter les résultats existants sur les réguliers
- Transformation de pCTRS
  - Explorer d'autres stratégies
- Transformation d'un programme logique en CS-programme avec sur-approximation
  - Améliorer la précision des approximations

# Remarques et questions

Merci de votre attention