# Utilisation de la sur-réduction pour le calcul du différentiel entre deux politiques de sécurité

Didier Villevalois

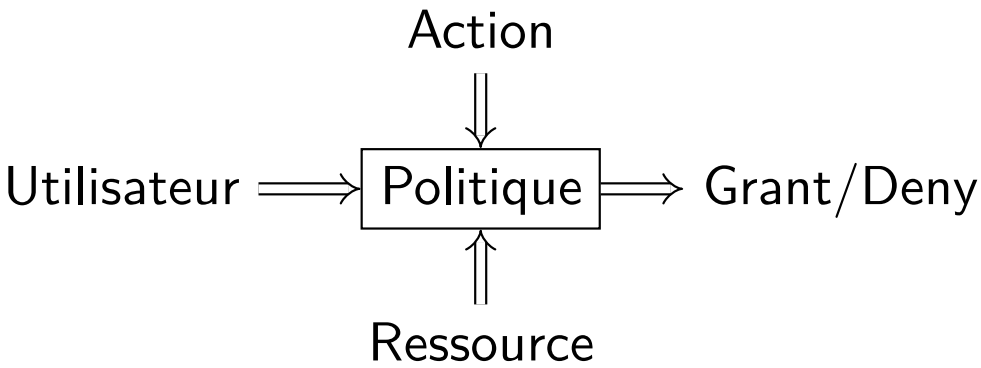
Encadrants
Clara Bertolissi et Jean-Marc Talbot

Lundi 22 Juin 2015

LABORATOIRE D'INFORMATIQUE FONDAMENTALE de Marseille
www.lif.univ-mrs.fr

Aix*Marseille université

# Politiques de contrôle d'accès




*Alice* a-t-elle le droit d'*éditer* la *base de donnée comptable* ?

# Réécriture de termes

Système de réécriture

$(\mathcal{F}, R)$ où $\mathcal{F}$ est une signature et $R$ est un ensemble de règles

Règle de réécriture

$l \to r$ telle que $l \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \setminus \mathcal{X}$ et $r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$

$$(append_1) \quad append(head :: tail, list) \to head :: append(tail, list)$$
$$(append_2) \quad append(\mathsf{nil}, list) \to list$$

Induit une relation $\to$ des termes vers les termes

Sur-réduction : généralisation de la réécriture

# Différentiel entre deux politiques de contrôle d'accès

Problème de la maintenance des politiques de contrôle d'accès

Modifier l'implémentation d'une politique et

- préserver son comportement global (préservation)
- préserver son comportement sur la partie existante (non-régression)

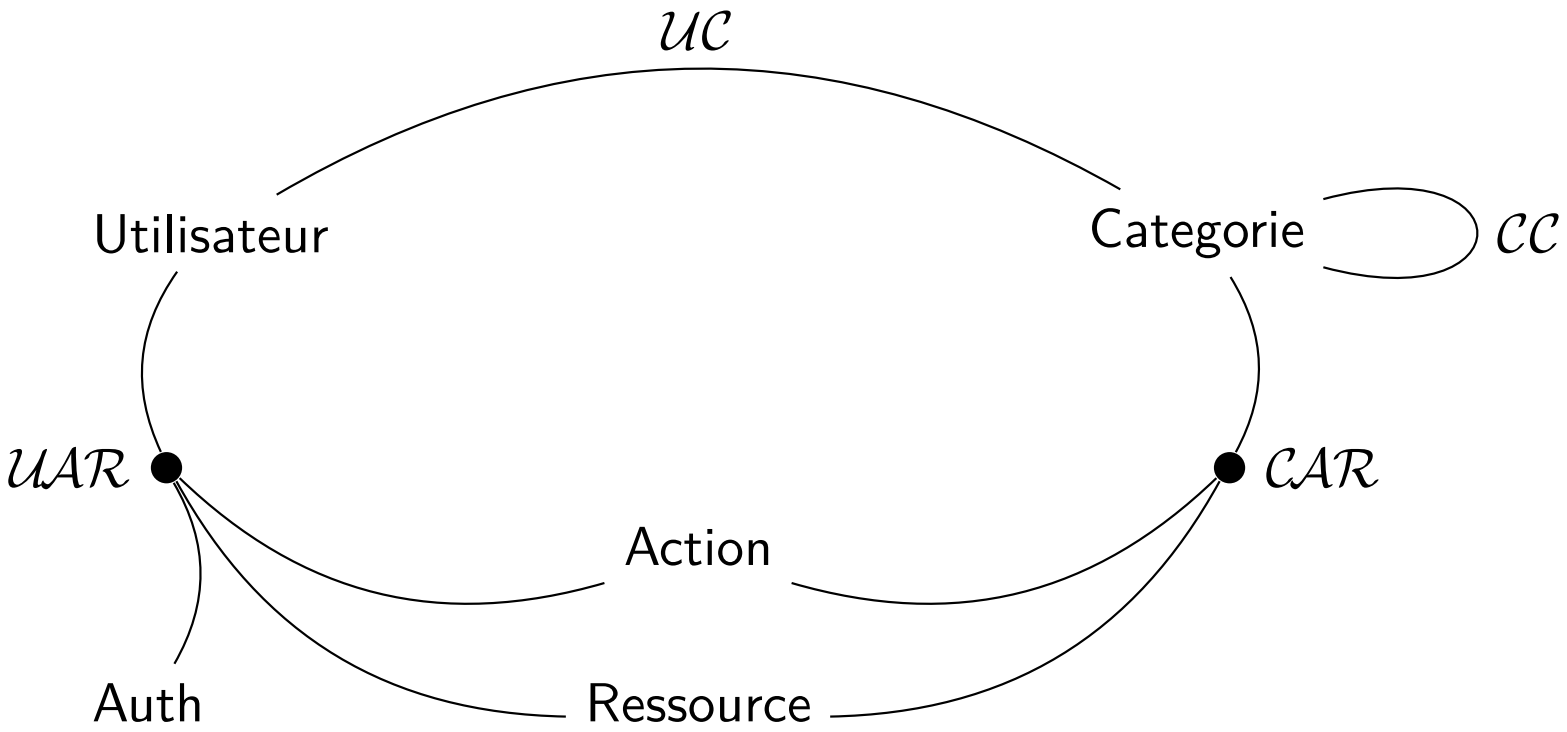
« Différentiel entre deux politiques »
Ensemble de contre-exemples à l'équivalence

1. Politiques de contrôle d'accès
   - Contrôle d'accès et réécriture de termes
   - Contrôle d'accès et sur-réduction

2. Différentiels de sur-réduction
   - Sur un même ensemble de constructeurs (préservation)
   - Sur deux ensembles de constructeurs (non-régression)

# Modèle $\mathcal{M}$ [Bar09]

Plusieurs modèles, tous exprimables dans le modèle générique $\mathcal{M}$




$$\forall u \in \mathcal{U}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}, \forall c \in \mathcal{C},$$
$$(u, c) \in \mathcal{UC} \wedge (\exists c', (c, c') \in \mathcal{CC} \wedge (c', a, r) \in \mathcal{CAR})$$
$$\Rightarrow (u, a, r, \mathsf{grant}) \in \mathcal{UAR}$$

# Spécifications formelles basées sur la réécriture [BF10]

$$
\begin{array}{lrcl}
(\mathrm{car}_1) & car(\mathsf{Admin}) & \to & (\mathsf{Edit}, \mathsf{PasswdFile}) :: (\mathsf{View}, \mathsf{PasswdFile}) :: \mathsf{nil} \\
(\mathrm{car}_2) & car(\mathsf{Accounting}) & \to & (\mathsf{Edit}, \mathsf{AccountDB}) :: (\mathsf{View}, \mathsf{SalesDB}) :: \mathsf{nil} \\
(\mathrm{car}_3) & car(\mathsf{Sales}) & \to & (\mathsf{Edit}, \mathsf{SalesDB}) :: (\mathsf{View}, \mathsf{AccountDB}) :: \mathsf{nil} \\
(\mathrm{cc}_1) & cc(\mathsf{Admin}) & \to & \mathsf{Accounting} :: \mathsf{Sales} :: \mathsf{nil} \\
(\mathrm{cc}_2) & cc(\mathsf{Accounting}) & \to & \mathsf{nil} \\
(\mathrm{cc}_3) & cc(\mathsf{Sales}) & \to & \mathsf{nil} \\
(\mathrm{uc}_1) & uc(\mathsf{Alice}) & \to & \mathsf{Admin} :: \mathsf{nil} \\
(\mathrm{uc}_2) & uc(\mathsf{Bob}) & \to & \mathsf{Sales} :: \mathsf{nil} \\
(\mathrm{uc}_3) & uc(\mathsf{Carol}) & \to & \mathsf{Accounting} :: \mathsf{nil}
\end{array}
$$

$$
\begin{array}{llll}
(\mathrm{car}_1) & car^*(h :: t) \to append(car(h), car^*(t)) & (\mathrm{car}_2) & car^*(\mathsf{nil}) \to \mathsf{nil} \\
(\mathrm{cc}_1^*) & cc^*(h :: t) \to append(h :: cc(h), cc^*(t)) & (\mathrm{cc}_2^*) & cc^*(\mathsf{nil}) \to \mathsf{nil}
\end{array}
$$

$$
\begin{array}{ll}
(\mathrm{uar}) & uar(utilisateur, action, ressource) \\
& \quad \to if((action, ressource) \in car^*(cc^*(uc(utilisateur))), \mathsf{grant}, \mathsf{deny})
\end{array}
$$

# Résolution d'une requête d'accès par réécriture

$\underline{uar(\mathsf{Alice}, \mathsf{Edit}, \mathsf{AccountDB})}$

$\to_{uar}$ $if((\mathsf{Edit}, \mathsf{AccountDB}) \in car^*(cc^*(\underline{uc(\mathsf{Alice})})), \mathsf{grant}, \mathsf{deny})$

$\to_{uc_1}$ $if((\mathsf{Edit}, \mathsf{AccountDB}) \in car^*(\underline{cc^*(\mathsf{Admin}::\mathsf{nil})}), \mathsf{grant}, \mathsf{deny})$

$\to_{cc^*_1}$ $if((\mathsf{Edit}, \mathsf{AccountDB}) \in \underline{car^*(\mathsf{Admin}::cc(\mathsf{Admin}):::cc^*(\mathsf{nil}))}, \mathsf{grant}, \mathsf{deny})$

$\to_{car^*_1}$ $if((\mathsf{Edit}, \mathsf{AccountDB}) \in \underline{car(\mathsf{Admin})}:::car^*(cc(\mathsf{Admin}):::cc^*(\mathsf{nil})), \mathsf{grant}, \mathsf{deny})$

$\vdots$

$\to_{\approx}$ $if(\underline{\mathsf{true} \vee (\mathsf{Edit}, \mathsf{AccountDB}) \in (\mathsf{View}, \mathsf{SalesDB})::\mathsf{nil}:::car^*(\mathsf{Sales}::\mathsf{nil}:::cc^*(\mathsf{nil}))}, \mathsf{grant}, \mathsf{deny})$

$\to_{\vee_1}$ $\underline{if(\mathsf{true}, \mathsf{grant}, \mathsf{deny})}$

$\to_{if_1}$ $\mathsf{grant}$

# Sur-réduction [MH94]

**Réécriture**, $t \to_{p,l\to r,\sigma} s$
variante $l \to r$ d'une règle
position non-variable $p$
substitution $\sigma$ tq $t|_p = \sigma(l)$
$s = t[\sigma(r)]_p$

**Sur-réduction**, $t \rightsquigarrow_{p,l\to r,\sigma} s$
variante $l \to r$ d'une règle
position non-variable $p$
unificateur $\sigma$ de $t|_p$ et $l$ (i.e. $\sigma(t|_p) = \sigma(l)$)
$s = \sigma(t[r]_p)$

$t =$ Sales $\in cc^*(\underline{uc(u)}) \approx$ true

Plusieurs choix de règle pour plusieurs résultats

- si $(\mathrm{uc}_1)$, $\sigma = \{u \mapsto \text{Alice}\}$ et $t \overset{*}{\rightsquigarrow}$ true
- si $(\mathrm{uc}_2)$, $\sigma = \{u \mapsto \text{Bob}\}$ et $t \overset{*}{\rightsquigarrow}$ true
- si $(\mathrm{uc}_3)$, $\sigma = \{u \mapsto \text{Carol}\}$ et $t \overset{*}{\rightsquigarrow}$ false

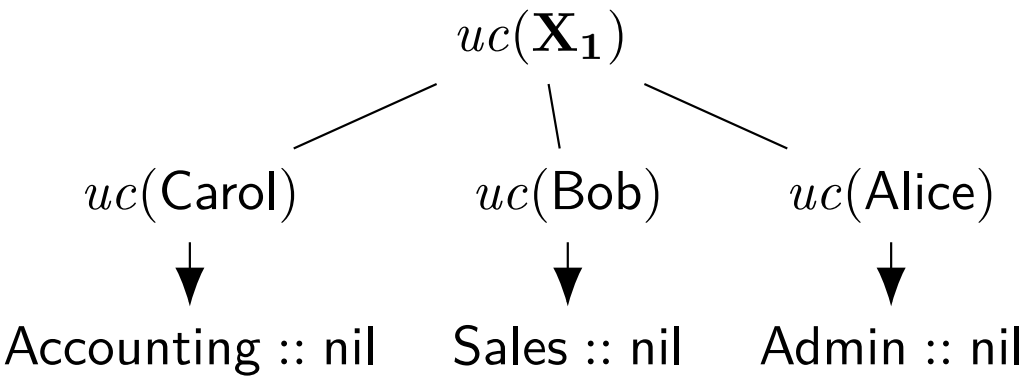
# Sur-réduction nécessaire la plus extérieure [AEH94]

Pour des systèmes à base de constructeurs

$$\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$$

Choix des sur-réductions possibles

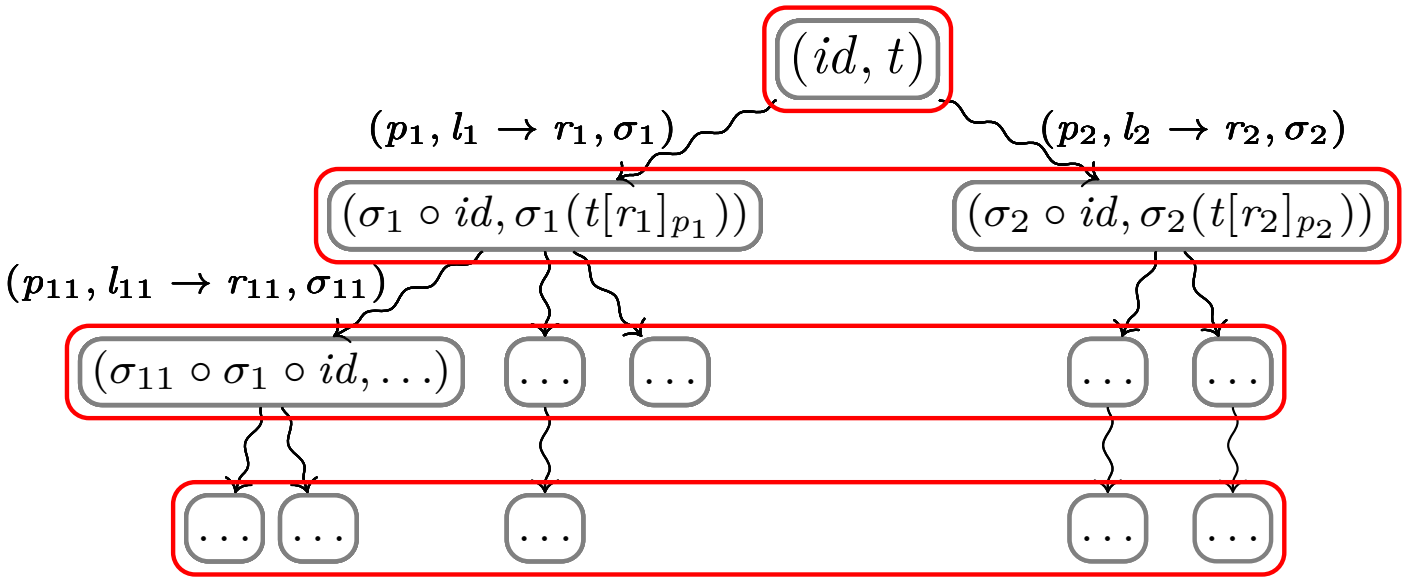
- en favorisant les sur-réductions *les plus extérieures*
- uniquement si elles sont *nécessaires*
- déterministiquement, par le parcours d'*arbres définitionnels* [Ant92]



# Énumération de solutions par sur-réduction

Stratégie : $t \mapsto \left\{(p, l \to r, \sigma) \mid t \rightsquigarrow_{p,l\to r,\sigma} \sigma(t[r]_p)\right\}$

Arbre de sur-réduction (nœuds $(substitution, terme)$)




Parcours en largeur d'abord (ensemble des nœuds à profondeur $n$)
Sur-réductions récursivement énumérables

# Interrogation par sur-réduction

$uar(\text{Alice}, a, r) \approx \text{grant} \overset{*}{\leadsto}_{\{a \mapsto \text{Edit}, r \mapsto \text{AccountingDB}\}} \text{true}$

$uar(\text{Alice}, a, r) \approx \text{grant} \overset{*}{\leadsto}_{\{a \mapsto \text{View}, r \mapsto \text{SalesDB}\}} \text{true}$

Modification du système :

$cc(\text{Administrative}) \to \text{Accounting} :: \text{Sales} :: \text{nil}$

remplacée par

$cc(\text{Administrative}) \to \text{Sales} :: \text{nil}$

$uar(\text{Alice}, a, r) \approx \text{grant} \overset{*}{\leadsto}_{\{a \mapsto \text{Edit}, r \mapsto \text{AccountingDB}\}} \text{false}$

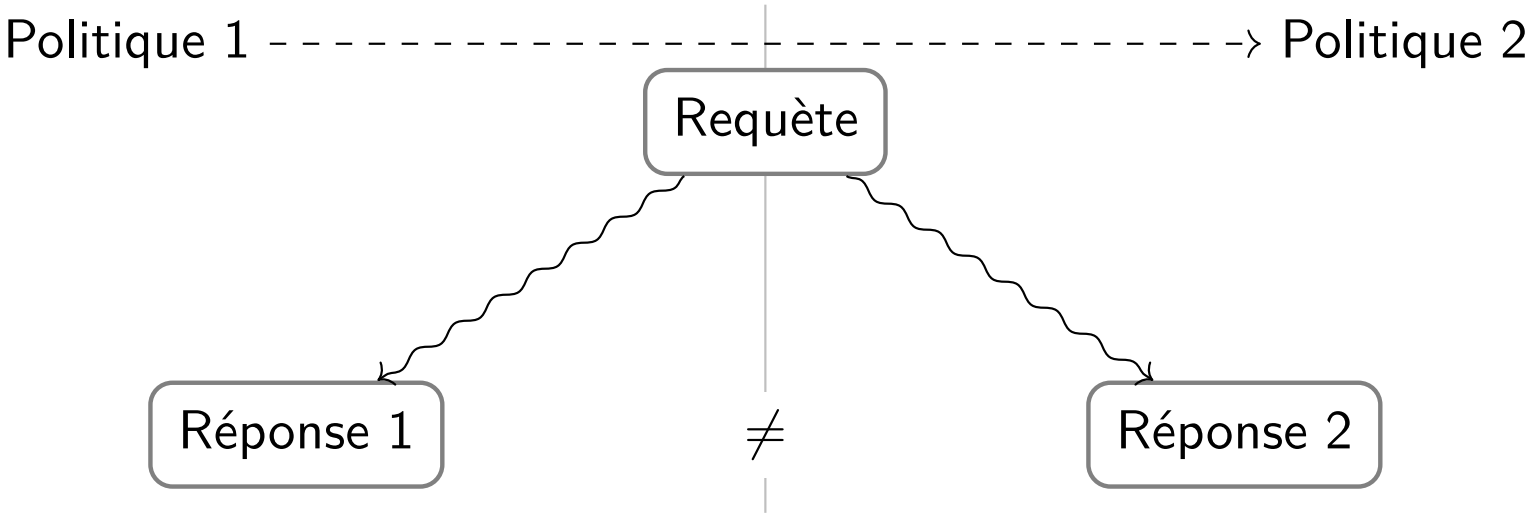
$uar(\text{Alice}, a, r) \approx \text{grant} \overset{*}{\leadsto}_{\{a \mapsto \text{View}, r \mapsto \text{SalesDB}\}} \text{false}$

$$(\text{uc}_1) \quad uc(\text{Alice}) \to \text{Admin} :: \text{nil}$$

$$(\text{car}_2) \quad car(\text{Accounting}) \to (\text{Edit}, \text{AccountDB}) :: (\text{View}, \text{SalesDB}) :: \text{nil}$$

# Différentiel de sur-réduction ?

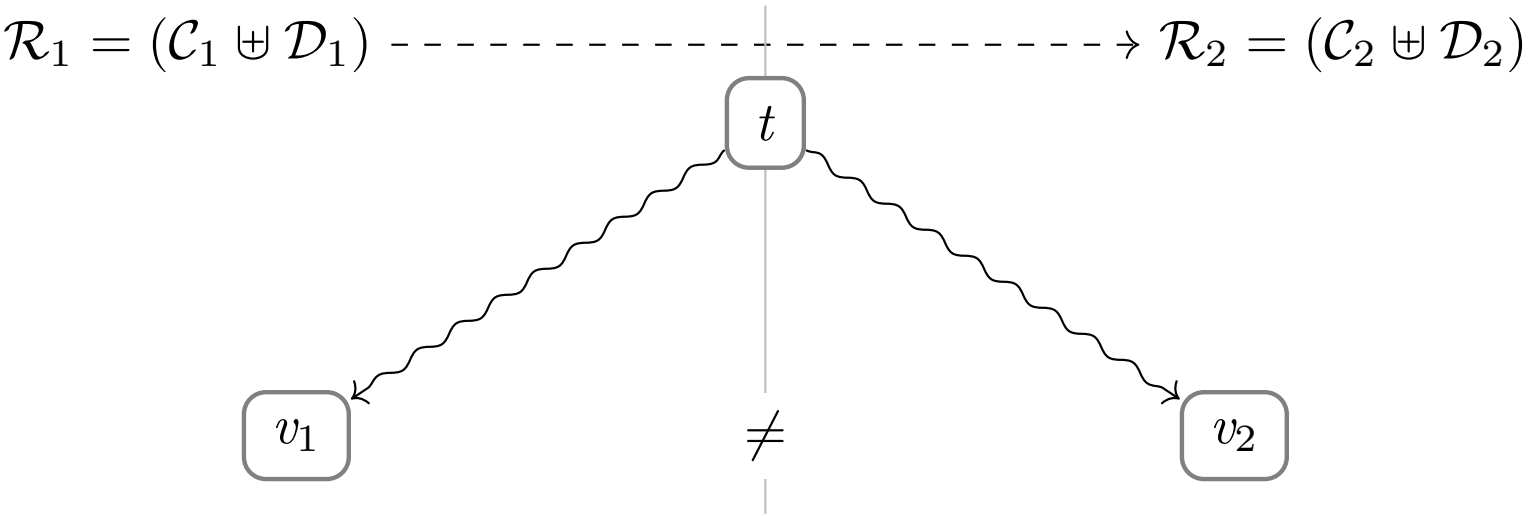
Déterminer en quoi deux politiques ont un comportement différent




- Préservation : Mêmes catégories, utilisateurs, actions et ressources
- Non-régression : Ajouts/Suppressions de catégories, utilisateurs, ...

# Différentiel de sur-réduction ?

Déterminer en quoi deux systèmes ont un comportement différent




- Préservation : Sur un même ensemble de constructeurs ($\mathcal{C}_1 = \mathcal{C}_2$)
- Non-régression : Sur deux ensembles de constructeurs ($\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$)

# Différentiel de sur-réduction ?

## Proposition ([AEH94])

*Étant donné un système* $\mathcal{R} = (\mathcal{F}, R)$ *et un terme* $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$

*Énumération par sur-réduction des valuations possibles de* $\mathcal{V}(t)$ *:*

$$\textit{si}\ \ t \approx x \overset{*}{\leadsto}_{\mathcal{R},\sigma} \text{true}\ \ \textit{alors}\ \ \sigma(t) \overset{*}{\to}_{\mathcal{R}} \sigma(x)$$

## Notre problème

Étant donnés deux systèmes $\mathcal{R}_1 = (\mathcal{F}_1, R_1)$ et $\mathcal{R}_2 = (\mathcal{F}_2, R_2)$
et un terme $t \in \mathcal{T}(\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{X})$

Énumération des valuations de $\mathcal{V}(t)$ pour lesquelles
on obtient des réductions différentes dans $\mathcal{R}_1$ et $\mathcal{R}_2$ ?

# Définition du problème de préservation ($\mathcal{C}_1 = \mathcal{C}_2$)

Deux systèmes $\mathcal{R}_1$ et $\mathcal{R}_2$ sur un même ensemble de constructeurs

**Définition (Différentiel de sur-réduction de $t$ dans $\mathcal{R}_2$ par rapport à $\mathcal{R}_1$)**

Soit un terme $t \in \mathcal{T}(\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{X})$.

$$\mathbf{D}_{\mathcal{R}_1,\mathcal{R}_2}(t) \stackrel{\text{def}}{=} \left\{(\sigma, v_1, v_2) \mid \sigma(t) \stackrel{*}{\to}_{\mathcal{R}_1} v_1 \text{ et } \sigma(t) \stackrel{*}{\to}_{\mathcal{R}_2} v_2 \text{ et } v_1 \neq v_2\right\}$$

Mise en algorithme :

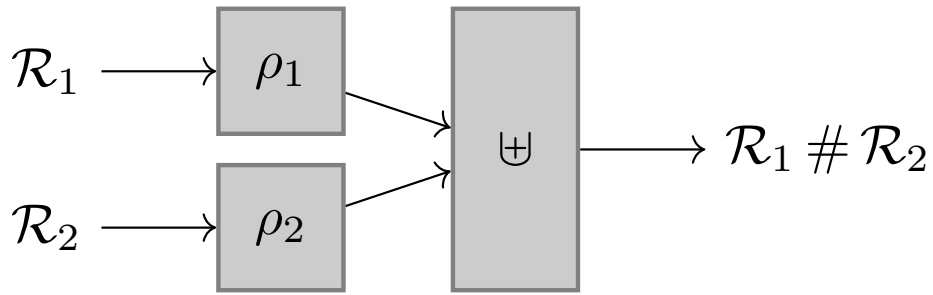
- Énumération des substitutions
- Réductions dans $\mathcal{R}_1$ et $\mathcal{R}_2$

Nombre non-borné de solutions, $\mathcal{R}_1$ ou $\mathcal{R}_2$ non-terminant :

- Nécessité d'entrelacer les étapes de calcul
- Donne un algorithme peu intelligible

# Transformation des systèmes

Idée : Adjoindre les deux systèmes en un seul




$\rho_k$ :

- Fonction bijective, $o \mapsto o_k$, pour $o \in \mathcal{D}$ et $k \in \{1, 2\}$
- Étendue naturellement $\rho_k$ aux termes et règles

$\uplus$ :

- Union disjointe de systèmes

# Calcul du différentiel

Pour un terme $t \in \mathcal{T}(\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{X})$
Sur-réduction dans $\mathcal{R}_1 \# \mathcal{R}_2$ du terme

$$\rho_1(t) \approx x \wedge \rho_2(t) \approx y \wedge x \not\approx y$$

## Theorème (Correction)

$(\rho_1(t) \approx x \wedge \rho_2(t) \approx y \wedge x \not\approx y) \overset{*}{\leadsto}_{\mathcal{R}_1 \# \mathcal{R}_2, \sigma}$ true
$\implies \quad (\sigma_{|\mathcal{V}(t)}, \sigma(x), \sigma(y)) \in \mathbf{D}_{\mathcal{R}_1, \mathcal{R}_2}(t)$

## Theorème (Complétude)

$(\sigma, v_1, v_2) \in \mathbf{D}_{\mathcal{R}_1, \mathcal{R}_2}(t)$
$\implies \quad (\rho_1(t) \approx x \wedge \rho_2(t) \approx y \wedge x \not\approx y) \overset{*}{\leadsto}_{\mathcal{R}_1 \# \mathcal{R}_2, \sigma'}$ true
où $\sigma' = \sigma \cup \{x \mapsto v_1, y \mapsto v_2\}[\mathcal{V}(t) \cup \{x, y\}]$

# Définition du problème de non-régression ($\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$)

Deux systèmes $\mathcal{R}_1$ et $\mathcal{R}_2$ sur deux ensembles de constructeurs

## Définition (Différentiel de sur-réduction de $t$ dans $\mathcal{R}_2$ par rapport à $\mathcal{R}_1$)

Soit un terme $t \in \mathcal{T}(\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{X})$.

$$\mathbf{D}^{-}_{\mathcal{R}_1,\mathcal{R}_2}(t) \stackrel{\text{def}}{=} \left\{(\sigma, v_1, \bot) \mid \sigma(t) \stackrel{*}{\rightarrow}_{\mathcal{R}_1} v_1 \text{ et } \sigma(t) \notin \mathcal{T}(\mathcal{C}_2 \uplus \mathcal{D}_2, \mathcal{X})\right\}$$

$$\mathbf{D}^{+}_{\mathcal{R}_1,\mathcal{R}_2}(t) \stackrel{\text{def}}{=} \left\{(\sigma, \bot, v_2) \mid \sigma(t) \stackrel{*}{\rightarrow}_{\mathcal{R}_2} v_2 \text{ et } \sigma(t) \notin \mathcal{T}(\mathcal{C}_1 \uplus \mathcal{D}_1, \mathcal{X})\right\}$$

$$\mathbf{D}^{\neq}_{\mathcal{R}_1,\mathcal{R}_2}(t) \stackrel{\text{def}}{=} \left\{(\sigma, v_1, v_2) \mid \sigma(t) \stackrel{*}{\rightarrow}_{\mathcal{R}_1} v_1 \text{ et } \sigma(t) \stackrel{*}{\rightarrow}_{\mathcal{R}_2} v_2 \text{ et } v_1 \neq v_2\right\}$$

$$\mathbf{D}^{\bot}_{\mathcal{R}_1,\mathcal{R}_2}(t) \stackrel{\text{def}}{=} \mathbf{D}^{-}_{\mathcal{R}_1,\mathcal{R}_2}(t) \cup \mathbf{D}^{\neq}_{\mathcal{R}_1,\mathcal{R}_2}(t) \cup \mathbf{D}^{+}_{\mathcal{R}_1,\mathcal{R}_2}(t)$$

# Transformation des systèmes

Ramener le problème de non-régression au problème de préservation

À partir de $\mathcal{R}_1 = (\mathcal{C}_1 \uplus \mathcal{D}_1, R_1)$ et $\mathcal{R}_2 = (\mathcal{C}_2 \uplus \mathcal{D}_2, R_2)$ :

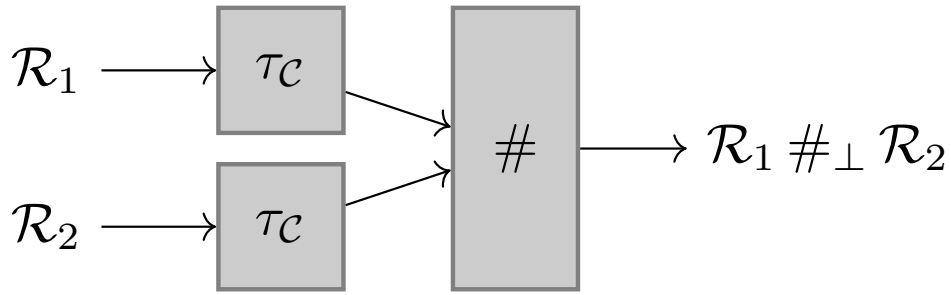
- $\mathcal{R}'_1$ défini sur $(\mathcal{C}_1 \cup \mathcal{C}_2 \cup \{\bot\}) \uplus \mathcal{D}_1$
  tq $\mathcal{R}'_1$ n'accepte que des termes sur $\mathcal{C}_1$, sinon réécrit vers $\bot$
- $\mathcal{R}'_2$ défini sur $(\mathcal{C}_1 \cup \mathcal{C}_2 \cup \{\bot\}) \uplus \mathcal{D}_2$
  tq $\mathcal{R}'_2$ n'accepte que des termes sur $\mathcal{C}_2$, sinon réécrit vers $\bot$

Transformation $\tau_{\mathcal{C}}$, où $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$

$$\mathbf{D}^{\bot}_{\mathcal{R}_1,\mathcal{R}_2}(t) = \mathbf{D}_{\tau_{\mathcal{C}}(\mathcal{R}_1),\tau_{\mathcal{C}}(\mathcal{R}_2)}(t)$$

# Transformation des systèmes

Idée : Exprimer $\#_\perp$ en fonction de $\tau_\mathcal{C}$ et $\#$




$\tau_\mathcal{C}$ :

- Envoie les arguments incorrectement valués vers $\perp$
- Pour chaque opération $o$, crée deux opérations
  - $o_0$, la copie de $o$
  - $o$, un opération de garde :
    $o(x_1, \ldots, x_n) \rightarrow if(\wedge_{i=1}^{n} valued(x_i), o_0(x_1, \ldots, x_n), \perp)$

Opération $valued(x)$ :

- true, si $x$ est dans l'ensemble initial de constructeur
- false, sinon

# Calcul du différentiel

Pour un terme $t \in \mathcal{T}(\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{X})$
Sur-réduction dans $\mathcal{R}_1 \#_\perp \mathcal{R}_2$ du terme

$$\rho_1(t) \approx x \wedge \rho_2(t) \approx y \wedge x \not\approx y$$

## Theorème (Correction)

$(\rho_1(t) \approx x \wedge \rho_2(t) \approx y \wedge x \not\approx y) \overset{*}{\leadsto}_{\mathcal{R}_1 \#_\perp \mathcal{R}_2, \sigma}$ true
$\implies \quad (\sigma_{|\mathcal{V}(t)}, \sigma(x), \sigma(y)) \in \mathbf{D}^{\perp}_{\mathcal{R}_1, \mathcal{R}_2}(t)$

## Theorème (Complétude)

$(\sigma, v_1, v_2) \in \mathbf{D}^{\perp}_{\mathcal{R}_1, \mathcal{R}_2}(t)$
$\implies \quad (\rho_1(t) \approx x \wedge \rho_2(t) \approx y \wedge x \not\approx y) \overset{*}{\leadsto}_{\mathcal{R}_1 \#_\perp \mathcal{R}_2, \sigma'}$ true
où $\sigma' = \sigma \cup \{x \mapsto v_1, y \mapsto v_2\}[\mathcal{V}(t) \cup \{x, y\}]$

# Conclusion

Le problème :

- Identification de contre-exemples à l'équivalence
  « Différentiels de sur-réduction »

La technique :

- Transformation des systèmes
- Sur-réduction

Pour deux systèmes $\mathcal{R}_1 = (\mathcal{F}_1, R_1)$ et $\mathcal{R}_2 = (\mathcal{F}_2, R_2)$
et un terme $t \in \mathcal{T}(\mathcal{F}_1 \cap \mathcal{F}_2, \mathcal{X})$

- Préservation ($\mathcal{C}_1 = \mathcal{C}_2$)
  $\mathbf{D}_{\mathcal{R}_1,\mathcal{R}_2}(t)$ est récursivement énumérable
- Non-régression ($\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$)
  $\mathbf{D}^{\perp}_{\mathcal{R}_1,\mathcal{R}_2}(t)$ est récursivement énumérable

# Perspectives

- Lever la restriction aux termes à racine opérationnelle [AEH94]
- Term Graph Rewriting pour les règles non-linéaires à droite [AEH94]
- Extension au Natural Narrowing [Esc03]

Merci !

# Réferences

[AEH94] S. Antoy, R. Echahed, and M. Hanus. A Needed Narrowing Strategy. In *Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 268–279. ACM, 1994. 00415.

[Ant92] S. Antoy. Definitional Trees. In *In Proc. of the 3rd International Conference on Algebraic and Logic Programming*, pages 143–157. Springer LNCS, 1992.

[Bar09] S. Barker. The next 700 access control models or a unifying meta-model ? In *SACMAT 2009, 14th ACM Symposium on Access Control Models and Technologies, Stresa, Italy, June 3-5, 2009, Proceedings*, pages 187–196, 2009.

[BF10] C. Bertolissi and M. Fernández. Category-Based Authorisation Models : Operational Semantics and Expressive Power. pages 140–156, 2010. 00002.

[Esc03] S. Escobar. Refining Weakly Outermost-Needed Rewriting and Narrowing. In *Proc. of 5th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'03*, pages 113–123. ACM Press, 2003. 00021.

[MH94] A. Middeldorp and E. Hamoen. Completeness Results for Basic Narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5(3-4) :213–253, 1994. 00138.