Réponse à des requêtes conjonctives en présence d'ontologies

Décidabilité, complexité et algorithmes

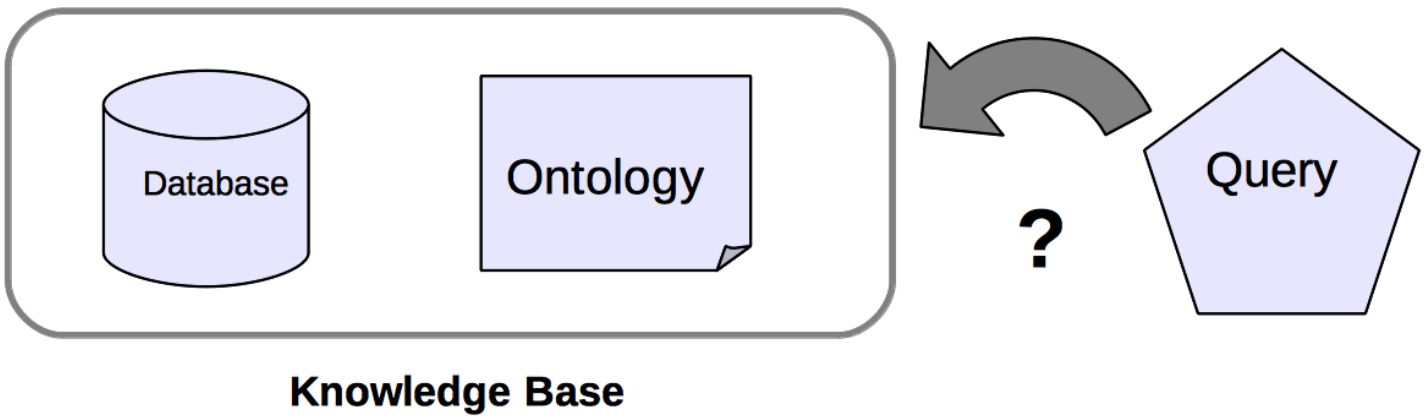
Michaël Thomazo
Encadré par Jean-François Baget et Marie-Laure Mugnier
Université Montpellier 2

30 juin 2014
RJCIA, Rouen

Vers une interrogation sémantique

- interrogation “intelligente” du Web (ou d’un sous-ensemble : Wikipedia par exemple) ;
- traitement sémantique de données bio-médicales.

Vue synthétique du problème



À la croisée de domaines

- représentation des connaissances et raisonnement
- base de données

Représentation des connaissances et raisonnement

- représenter des connaissances de manière formelle $\rightarrow$ ontologies
- ontologies : théorie logique représentant les connaissances d'un domaine
- principale approche pour exprimer des ontologies : les logiques de description
- raisonnement centré sur l'ontologie elle-même (historiquement)

Bases de données – un exemple

memberOf	
Alice	FencingClub
Bob	FencingClub
Craig	GardensFriends

SportClub
FencingClub

employeeOf	
Bob	LIRMM
Craig	_x1

$q_1(x) : -memberOf(x, y) \wedge SportClub(y)$

Bases de données – un exemple

memberOf	
Alice	FencingClub
Bob	FencingClub
Craig	GardensFriends

SportClub
FencingClub

employeeOf	
Bob	LIRMM
Craig	_x1

$q_1(x) : -memberOf(x, y) \wedge SportClub(y)$ {Alice, Bob}

Bases de données – un exemple

memberOf	
Alice	FencingClub
Bob	FencingClub
Craig	GardensFriends

SportClub
FencingClub

employeeOf	
Bob	LIRMM
Craig	_x1

$q_1(x) : -memberOf(x, y) \wedge SportClub(y)$ {Alice, Bob}

$q_2(x) : -memberOf(x, y) \wedge SportClub(y) \wedge hasSocialSecurityNumber(x, z)$

Bases de données – un exemple

memberOf	
Alice	FencingClub
Bob	FencingClub
Craig	GardensFriends

SportClub
FencingClub

employeeOf	
Bob	LIRMM
Craig	_x1

$q_1(x) : -memberOf(x, y) \wedge SportClub(y)$ {Alice, Bob}

$q_2(x) : -memberOf(x, y) \wedge SportClub(y) \wedge hasSocialSecurityNumber(x, z)$ $\emptyset$

Bases de données – un exemple

memberOf	
Alice	FencingClub
Bob	FencingClub
Craig	GardensFriends

SportClub
FencingClub

employeeOf	
Bob	LIRMM
Craig	_x1

$q_1(x) : -memberOf(x, y) \wedge SportClub(y)$ {Alice, Bob}

$q_2(x) : -memberOf(x, y) \wedge SportClub(y) \wedge hasSocialSecurityNumber(x, z)$ $\emptyset$

"Tout employé a un numéro de sécurité sociale."

Un problème à l'intersection des deux domaines

Ontology Based Query Answering (OBQA)

Interroger des données en prenant en compte une ontologie.

Outline

Panorama

Règles existentielles (1)

$$\forall \mathbf{x} \forall \mathbf{y} (B[\mathbf{x}, \mathbf{y}] \rightarrow \exists \mathbf{z} H[\mathbf{y}, \mathbf{z}]),$$

Une ontologie est un ensemble de règles existentielles.
Même forme logique que :

- TGDs [Abiteboul et al., 94]
- Datalog+/- [Calì et al., 09]
- règles de graphes conceptuels [Sowa 84, Mugnier and Salvat 96]

Règles existentielles (2)

- “value invention”
- pas de restrictions d’arité
- corps et tête de règles de forme quelconque
- généralise les logiques de description légères, couramment utilisées pour OBQA

Un exemple de règles existentielles

"Chaque employé a un numéro de sécurité sociale."

$$\forall x\ \forall y\ (\ \mathsf{employeeOf}(x, y) \rightarrow \exists z\ \mathit{hasSocialSecurityNumber}(x, z))$$

Le numéro de sécurité sociale n'est pas précisé – juste son existence.

Sur l'exemple

- données :
  $\text{memberOf}(\text{Alice}, \text{SportsCenter}) \wedge \ldots$
- ontologie :
  $\{\forall x\ \forall y\ (\ \text{employeeOf}(x, y) \rightarrow \exists z\ \mathit{hasSSN}(x, z)), \ldots\}$
- requête :
  $\exists x\ \exists y\ \exists z\ \mathit{memberOf}(x, y) \wedge \mathit{SportClub}(y) \wedge \mathit{hasSSN}(x, z)$

Des requêtes non-booléennes peuvent aussi être considérées.

Idée centrale

Se ramener à une “simple” requête SQL à évaluer sur une base de données.

Application de règles

$\forall x\ \forall y$ employeeOf(x,y) $\rightarrow \exists z$ *hasSocialSecurityNumber*(x, z)

memberOf(Alice, SportsCenter) $\wedge$
memberOf(Bob, SportsCenter) $\wedge$
memberOf(Craig, SwimmingCenter) $\wedge \ldots \wedge$
employeeOf(Bob,LIRMM) $\wedge$ employeeOf(Craig,_x1)

- $x \rightarrow$ Bob, $y \rightarrow$ *LIRMM* :
  création de hasSocialSecurityNumber(Bob,z_1)

Application de règles

$$\forall x\ \forall y\ \ \text{employeeOf(x,y)} \rightarrow \exists z\ \mathit{hasSocialSecurityNumber}(x, z)$$

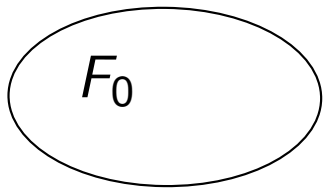
memberOf(Alice, SportsCenter) $\wedge$
memberOf(Bob, SportsCenter) $\wedge$
memberOf(Craig, SwimmingCenter) $\wedge \ldots \wedge$
employeeOf(Bob,LIRMM) $\wedge$ employeeOf(Craig,_x1)

- $x \rightarrow$ Bob, $y \rightarrow$ *LIRMM* :
  création de hasSocialSecurityNumber(Bob,z_1)
- $x \rightarrow$ Craig, $y \rightarrow$ _$x1$:
  création de hasSocialSecurityNumber(Craig,z_2)

z_1 et z_2 sont existentiellement quantifiées.

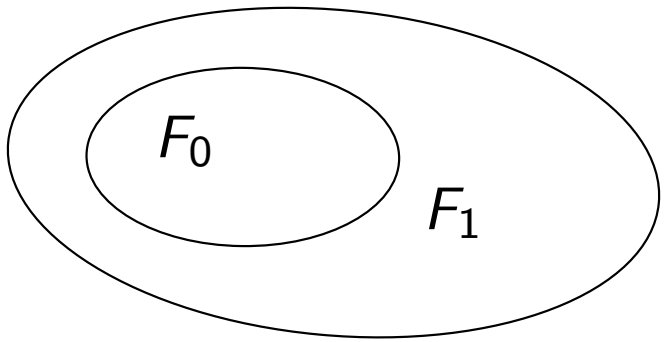
Modèle canonique (matérialisation)

Chase, marche avant : F et $\mathcal{R}$ en entrée



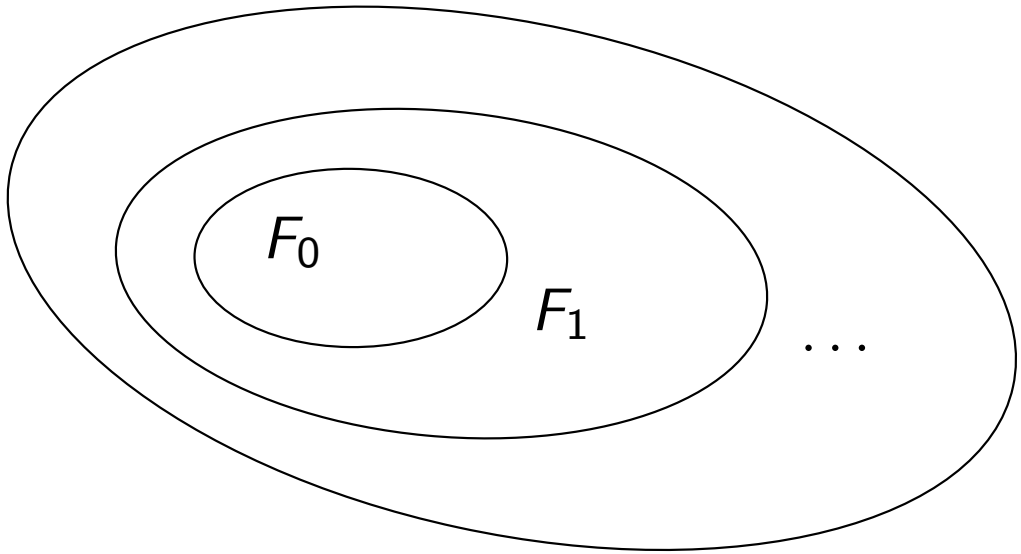
Modèle canonique (matérialisation)

Chase, marche avant : F et $\mathcal{R}$ en entrée



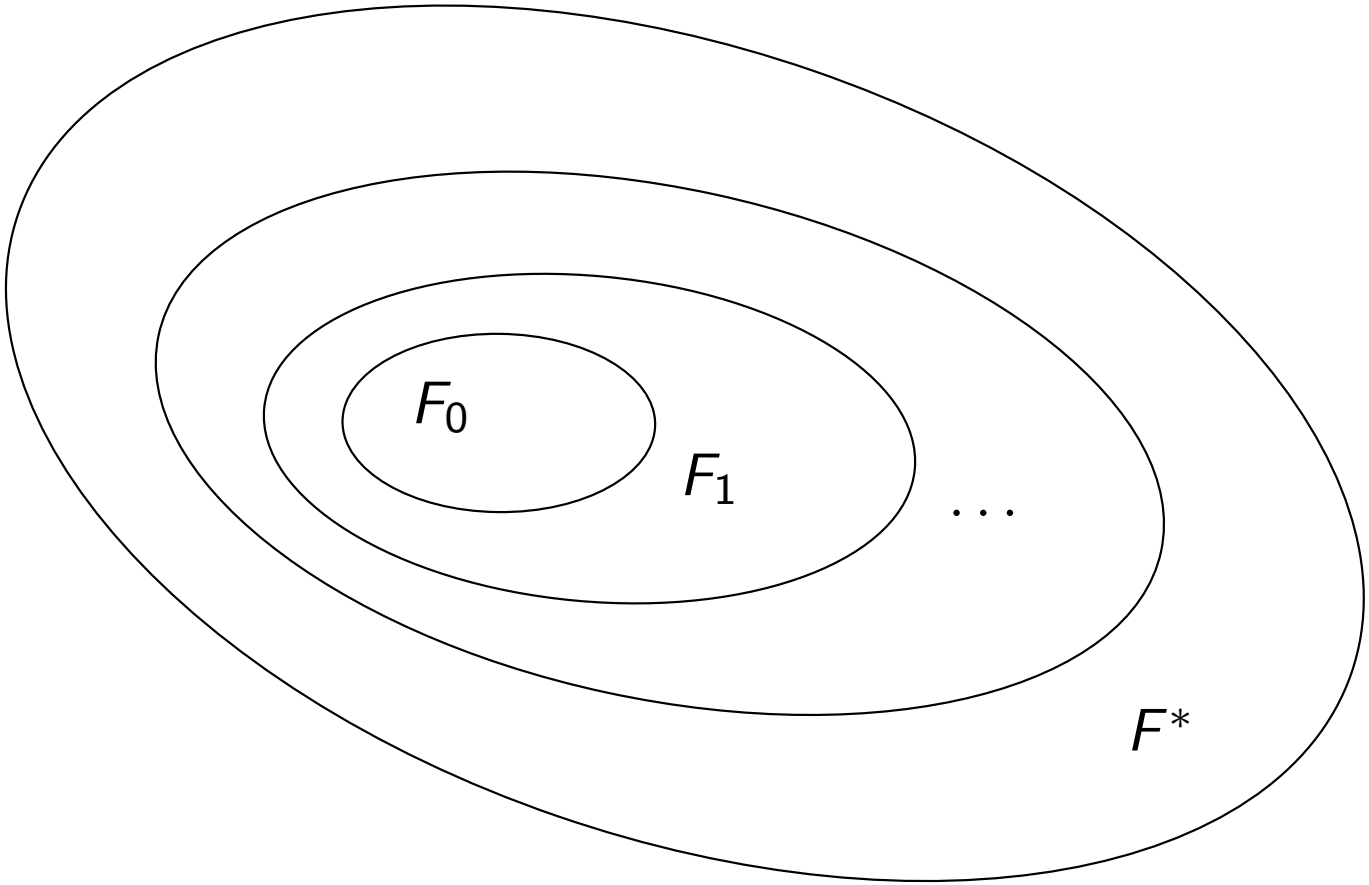
Modèle canonique (matérialisation)

Chase, marche avant : F et $\mathcal{R}$ en entrée



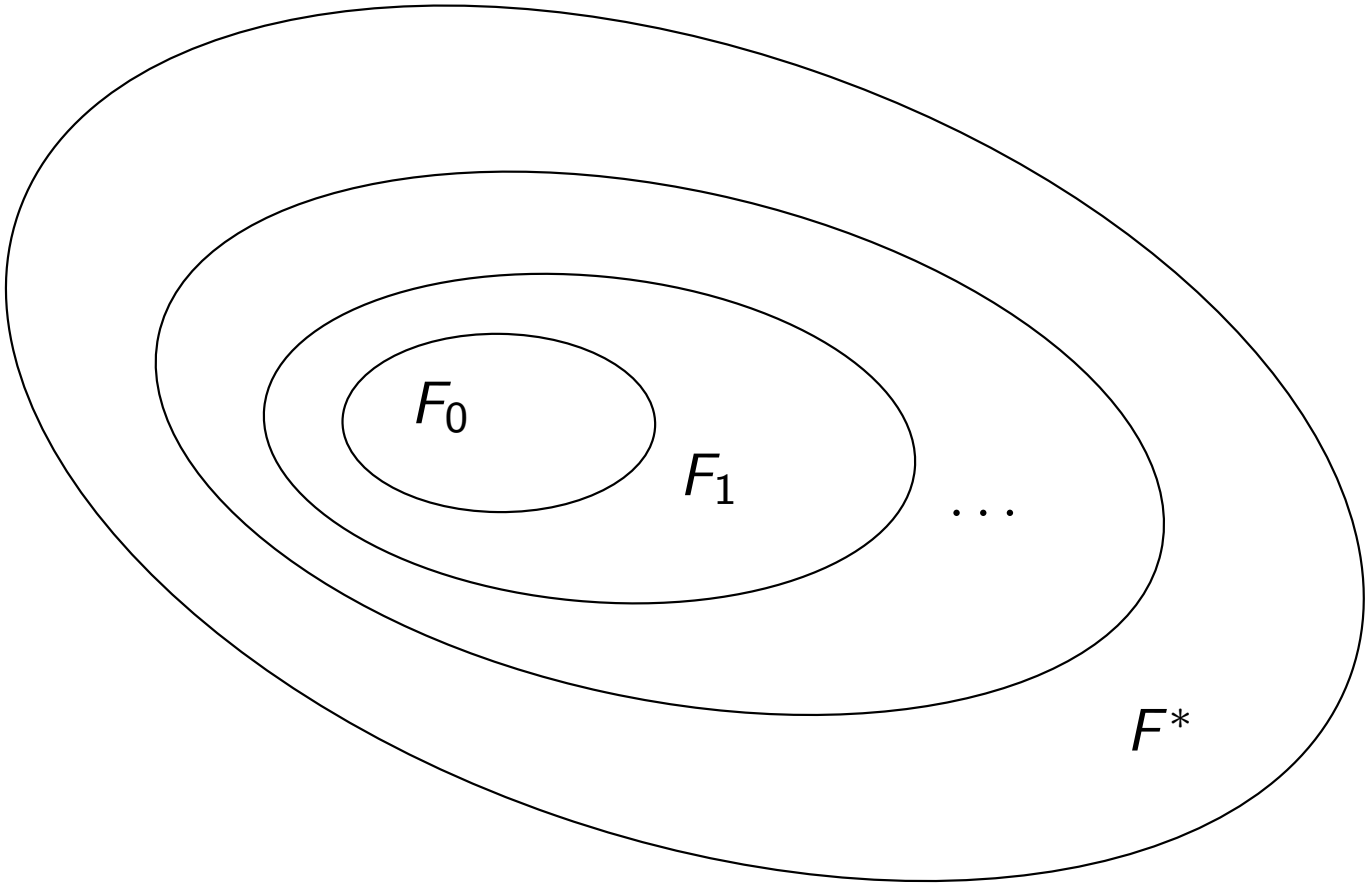
Modèle canonique (matérialisation)

Chase, marche avant : F et $\mathcal{R}$ en entrée



Modèle canonique (matérialisation)

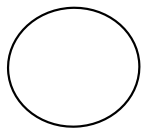
Chase, marche avant : F et $\mathcal{R}$ en entrée



Property : $\forall Q \ (F, \mathcal{R} \models Q \Leftrightarrow F^* \models Q)$

Sans matérialisation ?

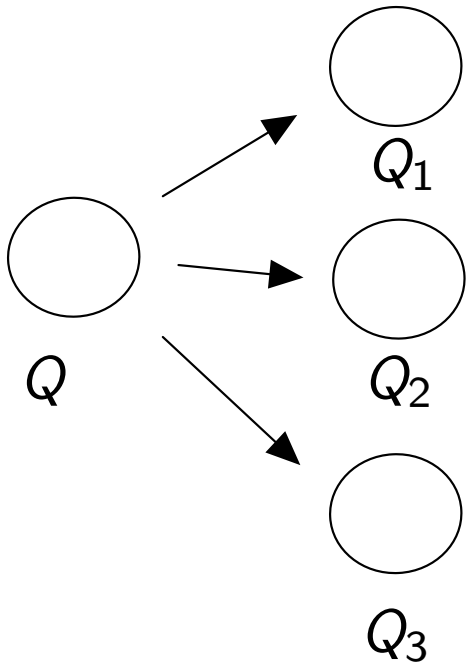
Réécriture, marche arrière : Q and $\mathcal{R}$ en entrée ([Calvanese et al., 2005] entre autres)



Q

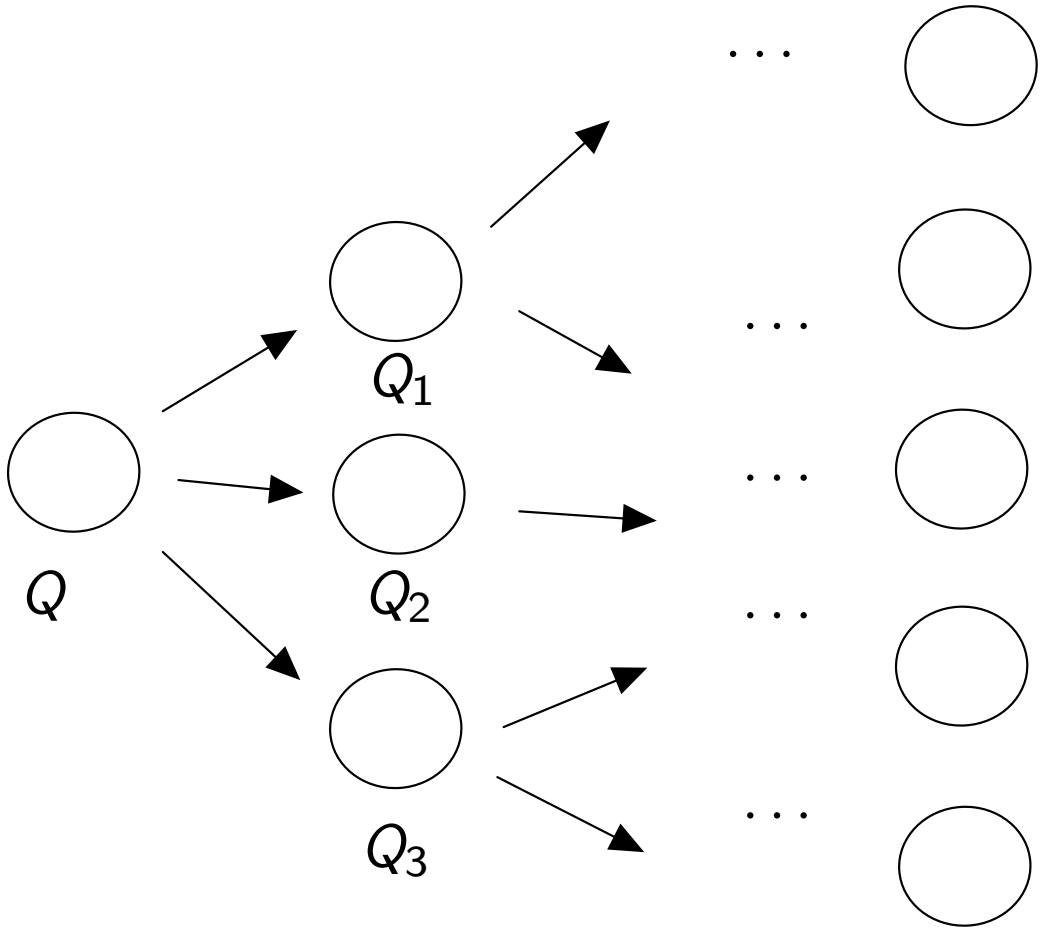
Sans matérialisation ?

Réécriture, marche arrière : Q and $\mathcal{R}$ en entrée ([Calvanese et al., 2005] entre autres)



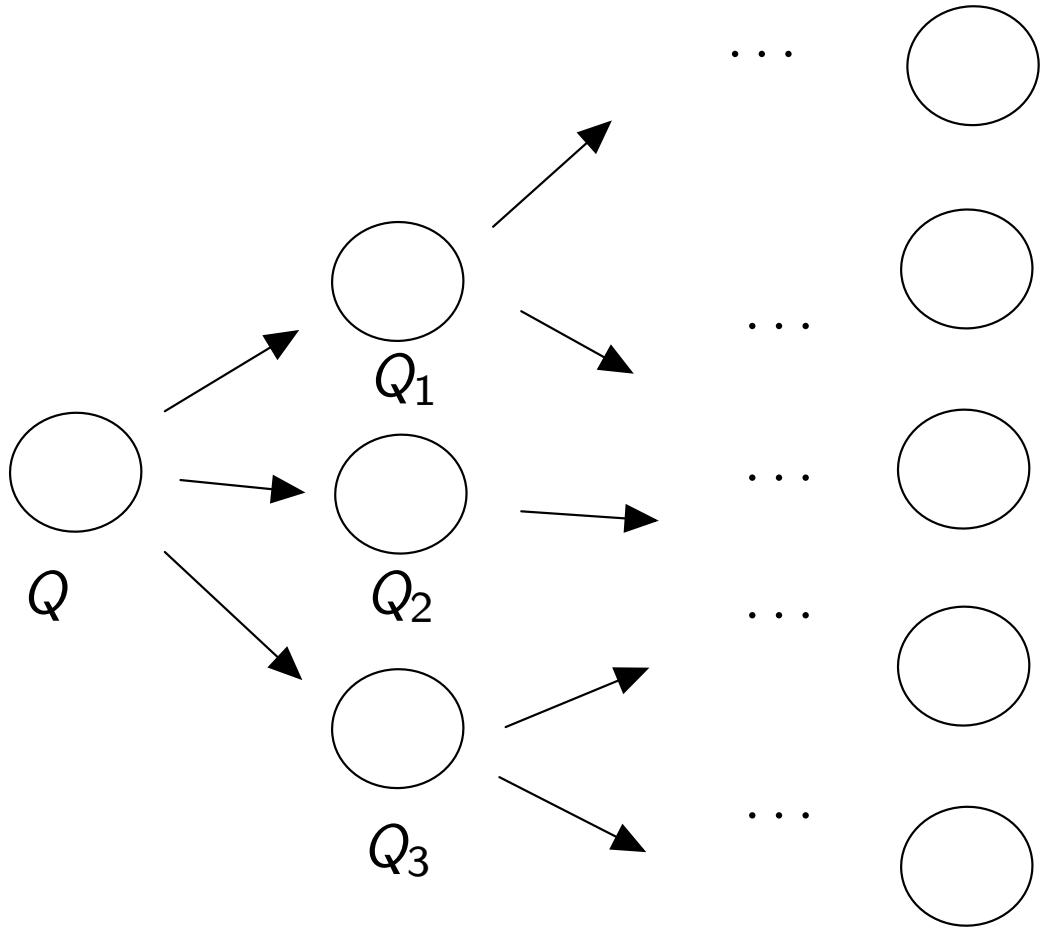
Sans matérialisation ?

Réécriture, marche arrière : Q and $\mathcal{R}$ en entrée ([Calvanese et al., 2005] entre autres)



Sans matérialisation ?

Réécriture, marche arrière : Q and $\mathcal{R}$ en entrée ([Calvanese et al., 2005] entre autres)



Propriété : $\forall F \ (F, \mathcal{R} \models Q \Leftrightarrow \exists Q_i : F \models Q_i)$

Indécidabilité

- le modèle canonique peut être infini :
  $\forall x\ (\ \mathsf{Human}(x) \rightarrow\ \mathsf{parent}(y, x) \wedge\ \mathsf{Human}(y))$
- une réécriture de taille finie peut ne pas exister :
  $\forall x\ \forall\ y\ (\ \mathsf{parent}(x, y) \wedge\ \mathsf{Human}(x) \rightarrow\ \mathsf{Human}(y))$

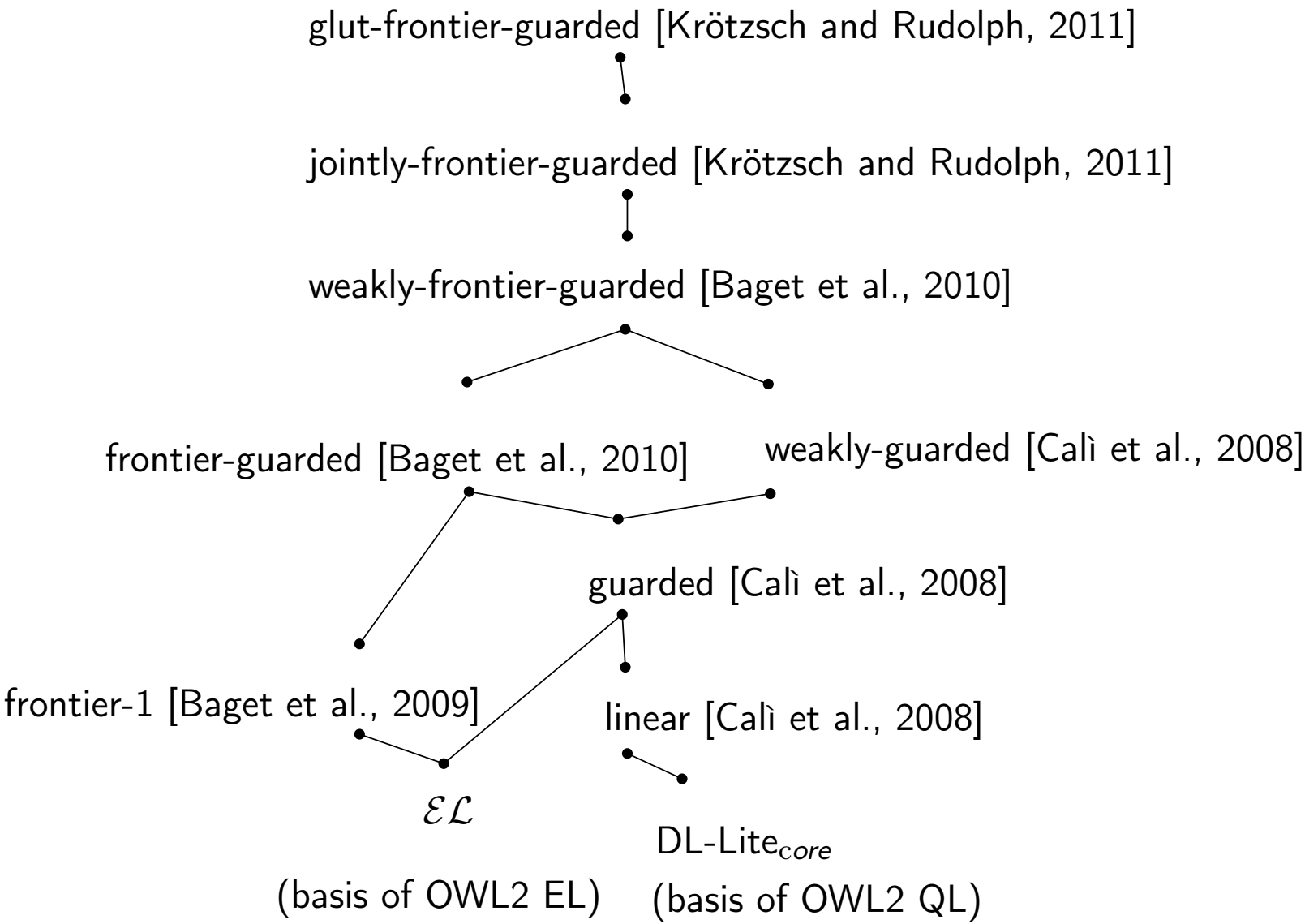
Indécidabilité [Beeri and Vardi 1981], entre autres

Répondre à des requêtes conjonctives en présence de règles existentielles n'est pas décidable.

Principaux critères de décidabilité

- quand le modèle canonique est équivalent à un fait fini ;
- quand il existe une réécriture finie ;
- quand le modèle canonique est “semblable” à un arbre (*bts* : bounded treewidth set).

Classes concrètes *bts* connues



Contributions

- définition d'une nouvelle classe décidable ;
- construction d'un algorithme optimal dans le pire des cas pour cette classe ;
- adaptation de l'algorithme à des sous-classes ;
- design, prototype et première évaluation d'un algorithme générique de réécriture.

Contributions

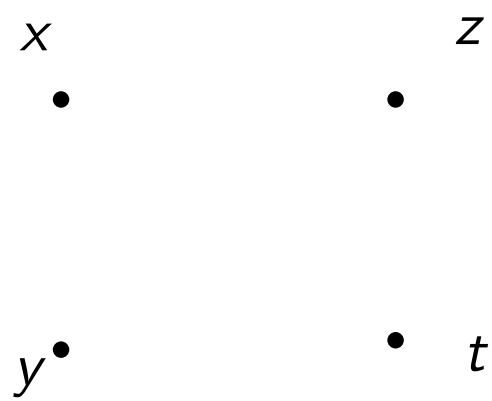
- définition d'une nouvelle classe décidable ;
- construction d'un algorithme optimal dans le pire des cas pour cette classe ;
- adaptation de l'algorithme à des sous-classes ;
- design, prototype et première évaluation d'un algorithme générique de réécriture.

Outline

Une nouvelle classe décidable

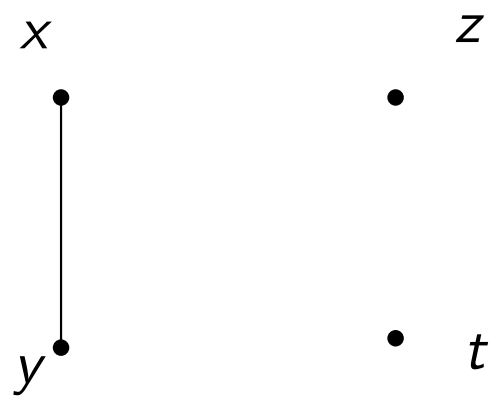
Graph primal d'un fait

Graphe primal de $\exists x \exists y \exists z \exists t \; r(x, y) \wedge p(x, z, t) \wedge r(y, z)$



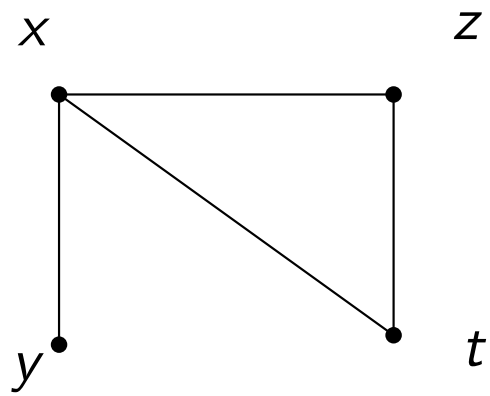
Graph primal d'un fait

Graphe primal de $\exists x \exists y \exists z \exists t \; r(x, y) \wedge p(x, z, t) \wedge r(y, z)$



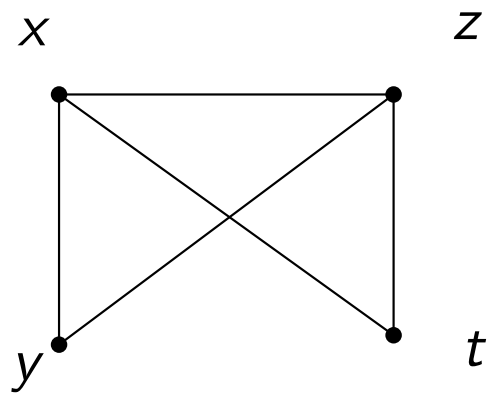
Graph primal d'un fait

Graphe primal de $\exists x \exists y \exists z \exists t \; r(x, y) \wedge p(x, z, t) \wedge r(y, z)$



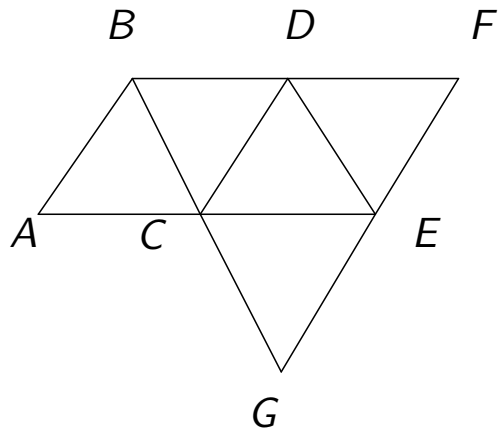
Graph primal d'un fait

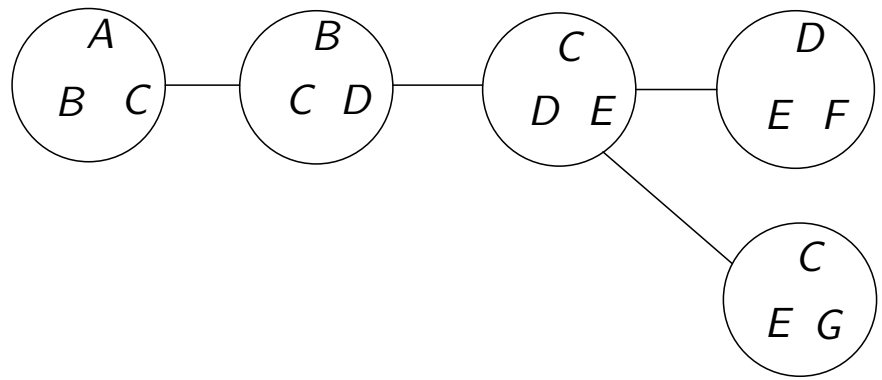
Graphe primal de $\exists x \exists y \exists z \exists t \; r(x, y) \wedge p(x, z, t) \wedge r(y, z)$



Décomposition et largeur arborescentes (1)

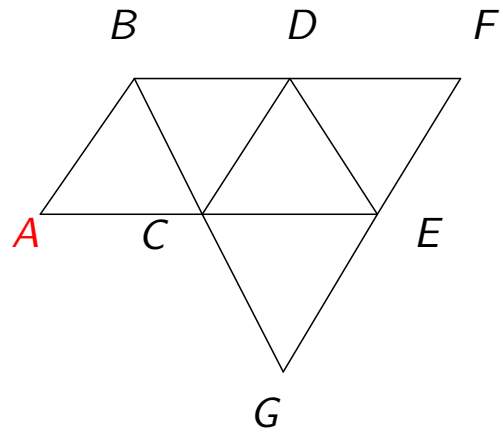
Robertson and Seymour, 1984

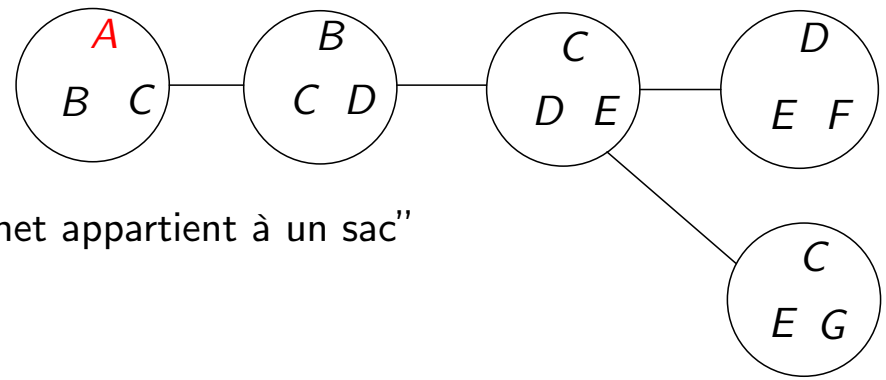




Décomposition et largeur arborescentes (1)

Robertson and Seymour, 1984

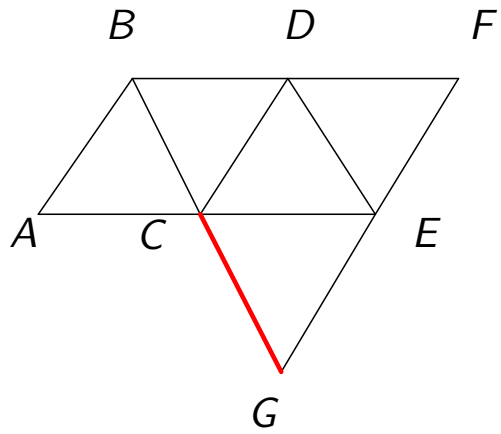


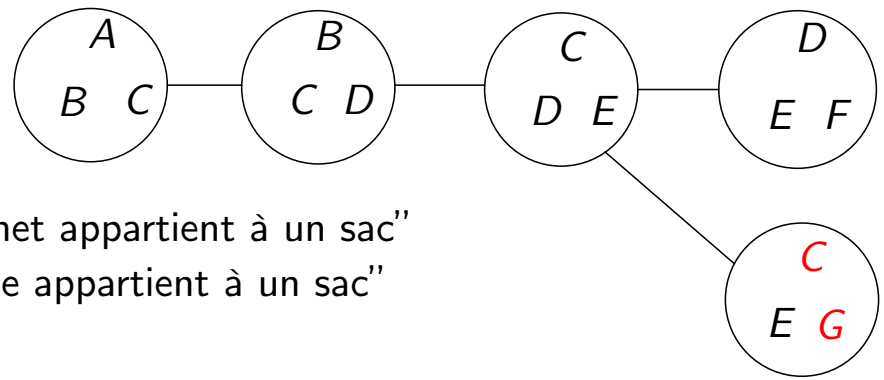


"Chaque sommet appartient à un sac"

Décomposition et largeur arborescentes (1)

Robertson and Seymour, 1984



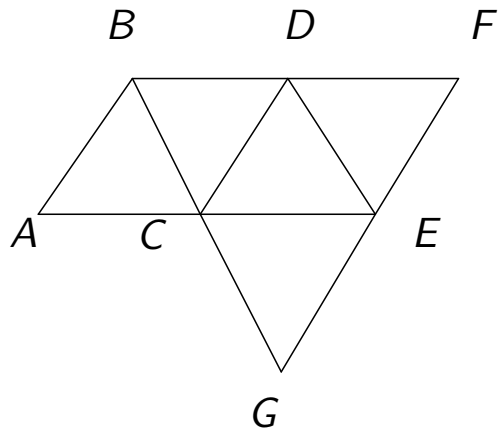


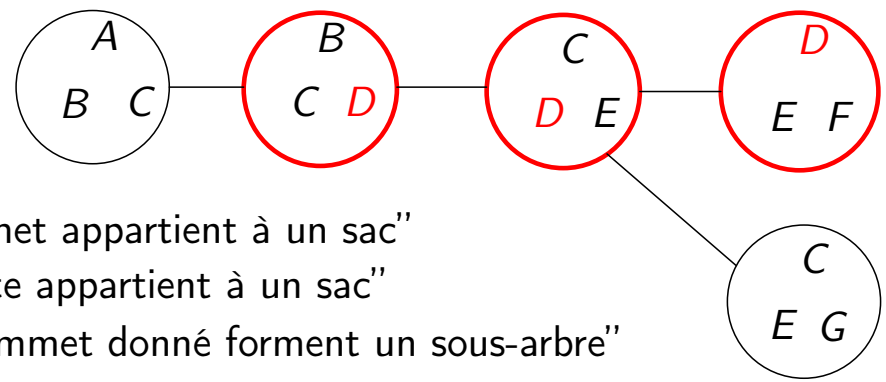
"Chaque sommet appartient à un sac"

"Chaque arête appartient à un sac"

Décomposition et largeur arborescentes (1)

Robertson and Seymour, 1984





"Chaque sommet appartient à un sac"

"Chaque arête appartient à un sac"

"Les sacs contenant un sommet donné forment un sous-arbre"

Décomposition et largeur arborescentes (2)

Treewidth

La largeur d'une décomposition arborescente est le nombre maximal de sommets dans un sac moins un. La largeur arborescente (treewidth) d'un graphe est la plus petite largeur d'une de ses décompositions arborescentes.

Treewidth d'un arbre

La treewidth d'un arbre est 1.

A quel point ressemble un graphe à un arbre ?

Décidabilité de *bts*

Calí et al. 2008, Baget et. al 2009

Treewidth d'un fait

La treewidth d'un fait est la treewidth de son graph primal.

Bounded treewidth sets (*bts*)

Un ensemble de règles est à treewidth bornée si pour tout fait F, la treewidth de son modèle canonique est bornée.

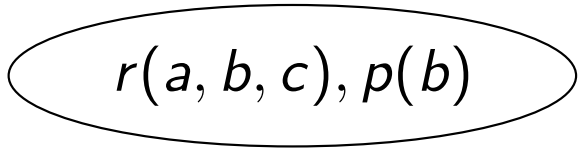
OBQA avec *bts* est décidable

Cela provient d'un résultat de Courcelle sur la logique monadique du second ordre (1990).

Pas de manière imposée d'obtenir une décomposition – pas d'algorithmes connus

Décomposition arborescente gloutonne

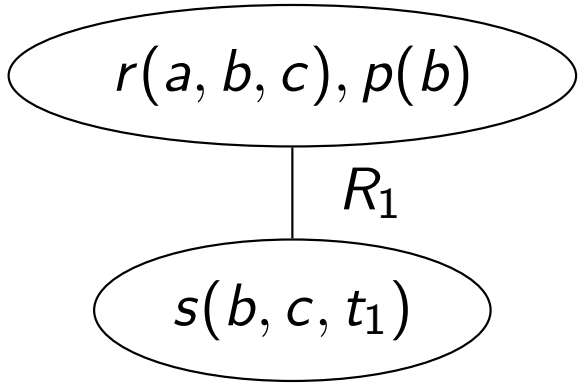
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

Décomposition arborescente gloutonne

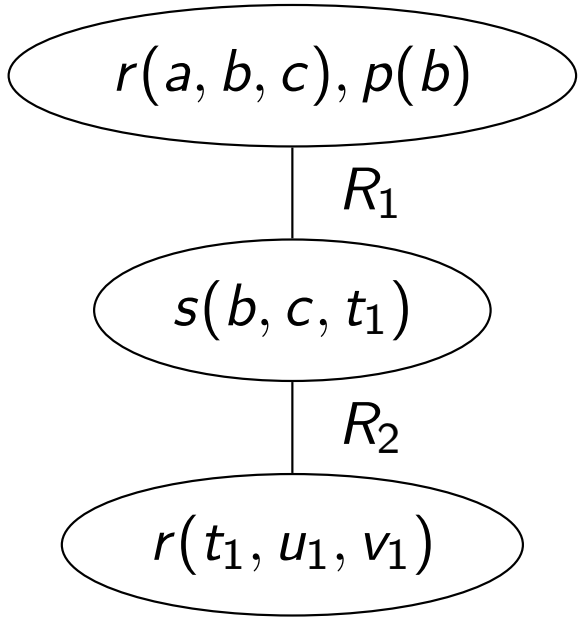
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

Décomposition arborescente gloutonne

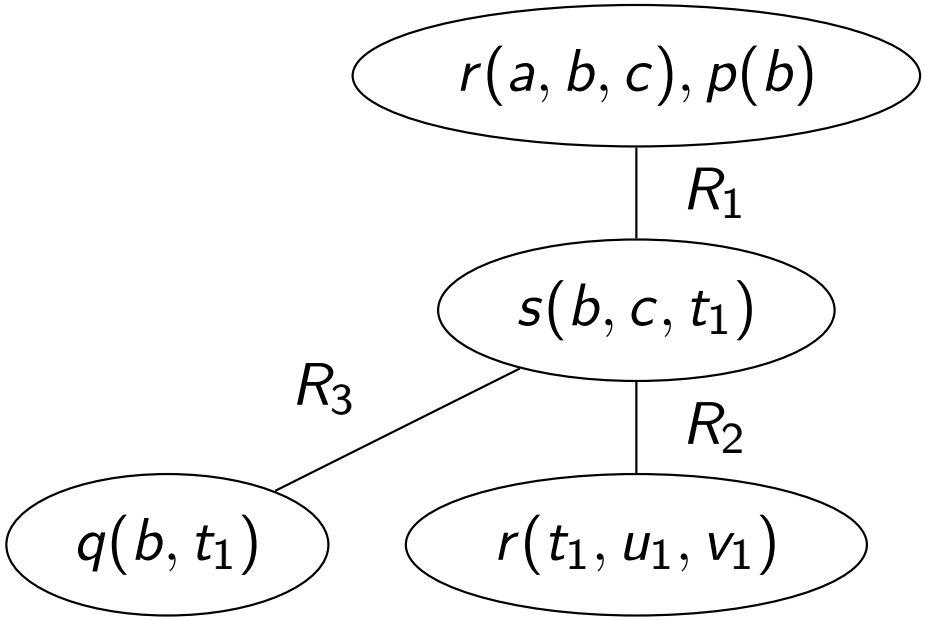
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

Décomposition arborescente gloutonne

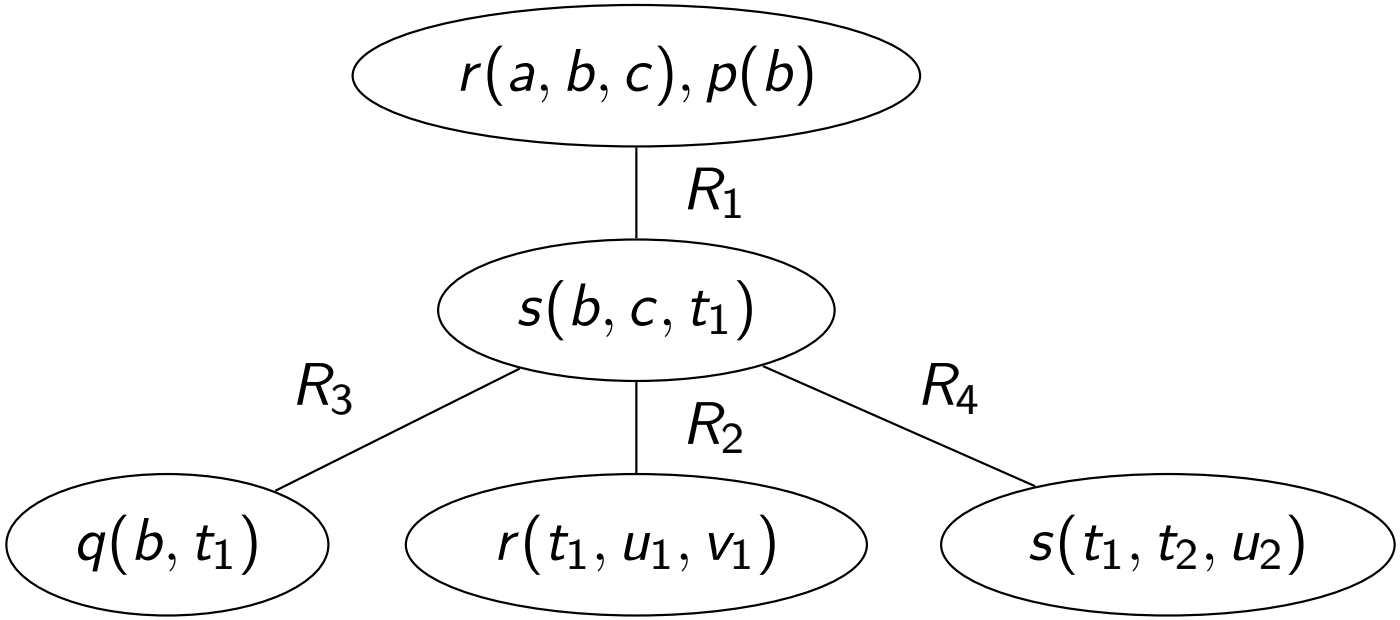
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

Décomposition arborescente gloutonne

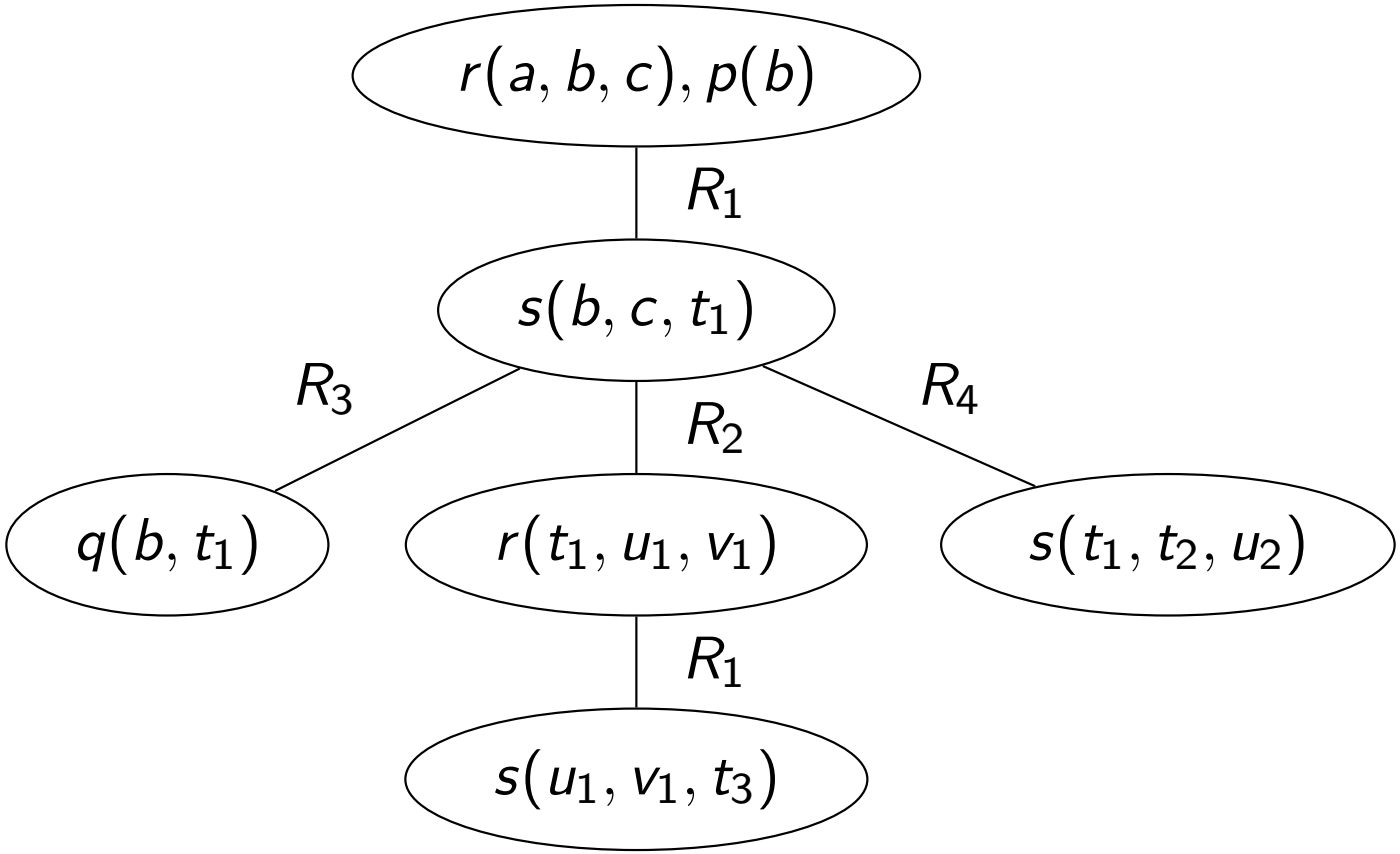
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

Décomposition arborescente gloutonne

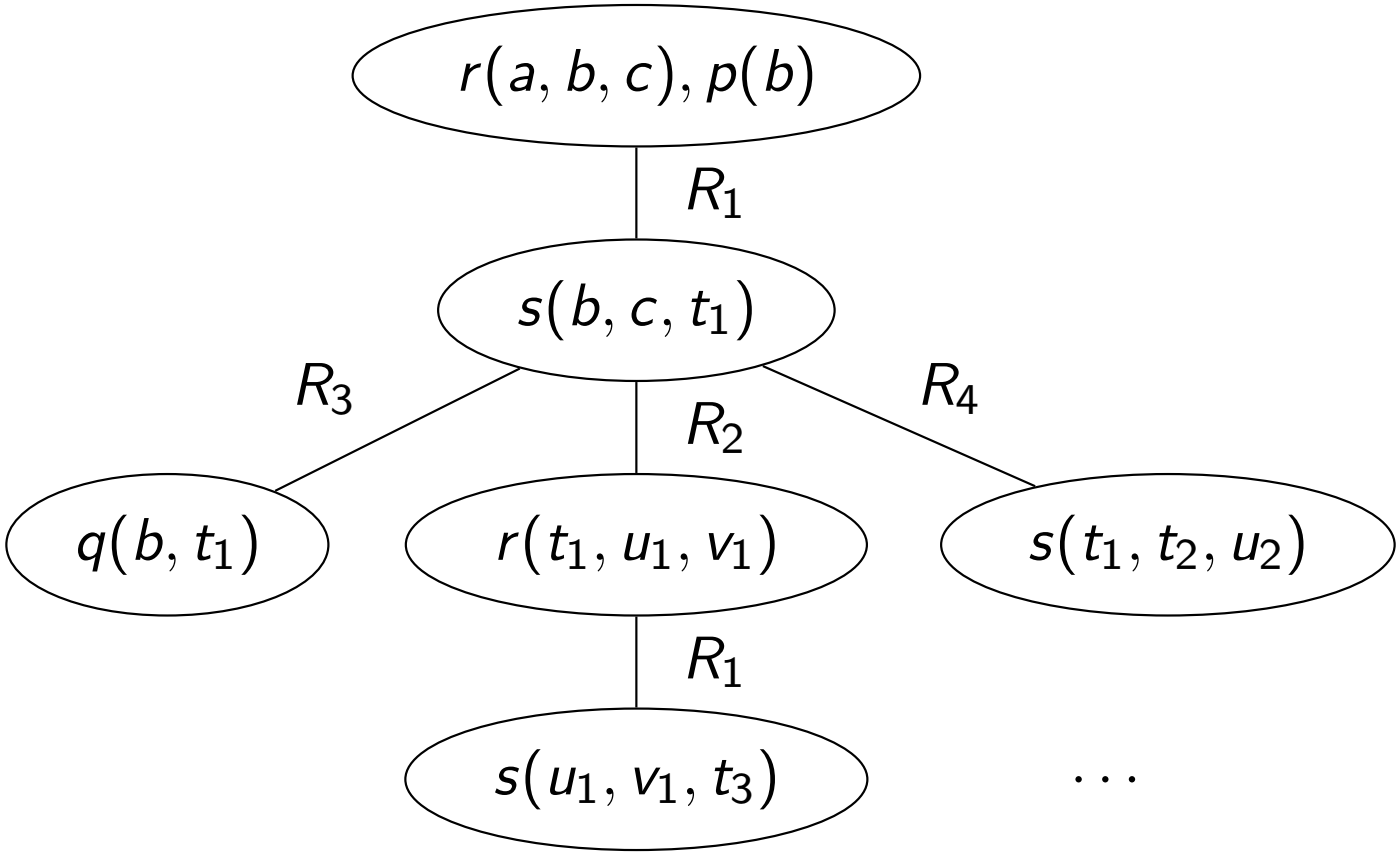
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

Décomposition arborescente gloutonne

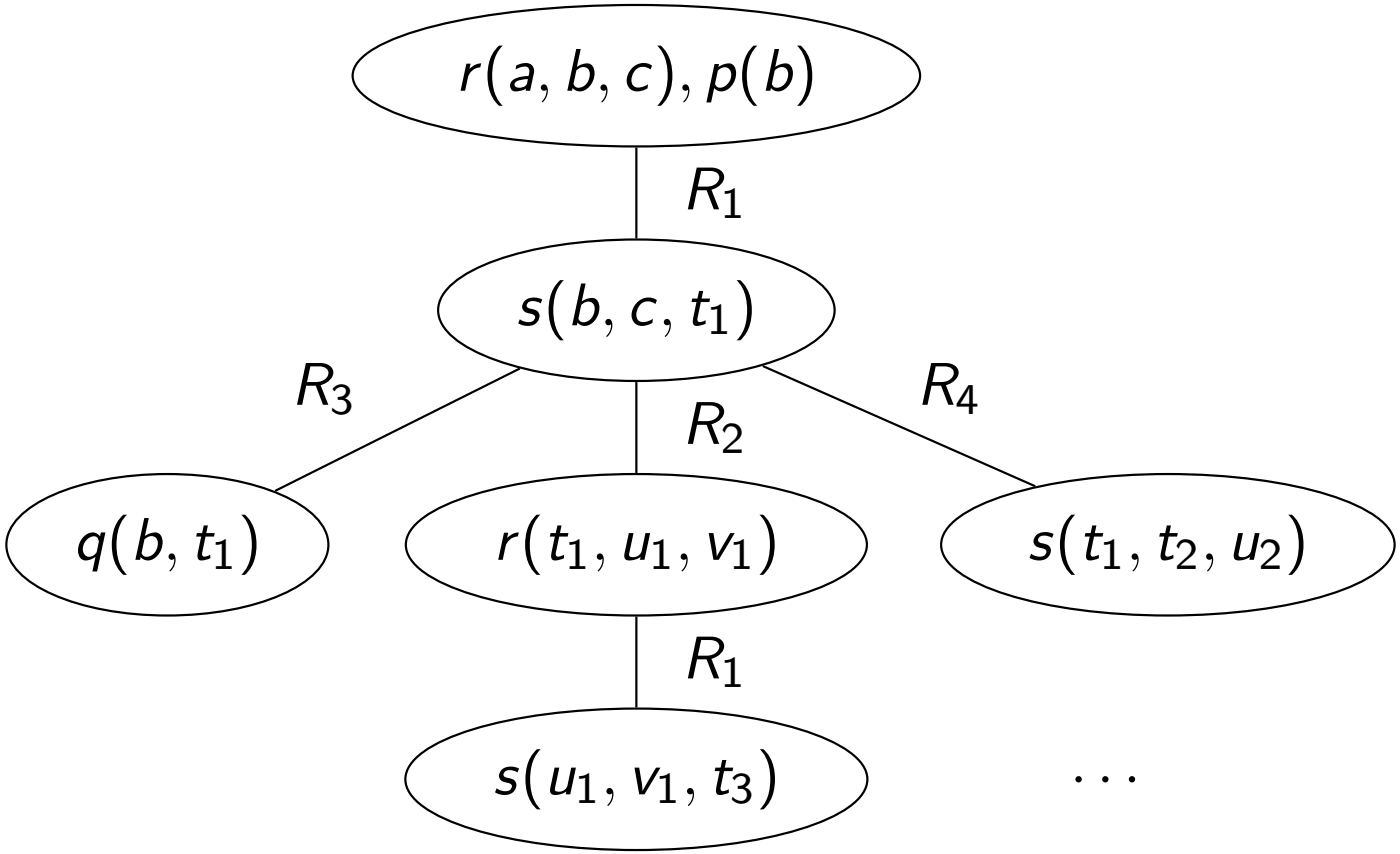
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

Décomposition arborescente gloutonne

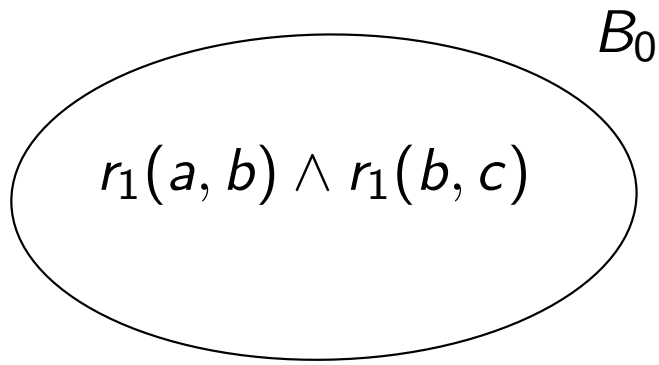
Baget, Mugnier, Rudolph, T., 2011



$R_1 = r(x, \mathbf{y}, \mathbf{z}) \rightarrow s(\mathbf{y}, \mathbf{z}, t)$	$R_2 = s(x, y, \mathbf{z}) \rightarrow r(\mathbf{z}, u, v)$
$R_3 = s(\mathbf{x}, y, \mathbf{z}) \wedge p(\mathbf{x}) \rightarrow q(\mathbf{x}, \mathbf{z})$	$R_4 = q(x, \mathbf{z}) \rightarrow s(\mathbf{z}, t, u)$

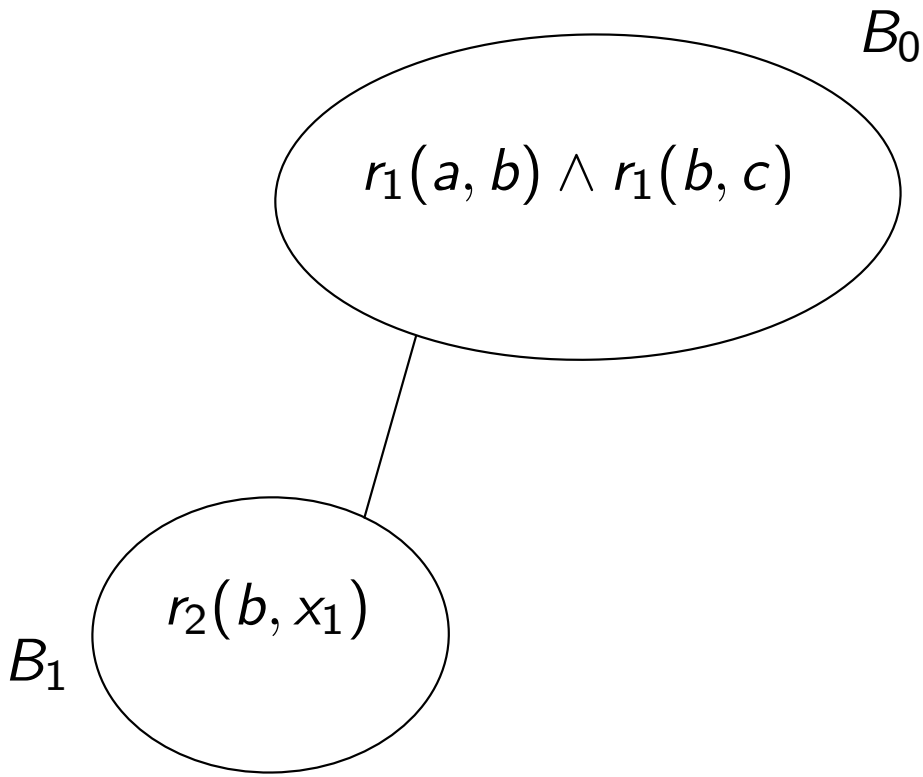
Ultimate derivation tree

Un exemple de règles qui est *bts* mais pas *gbts*



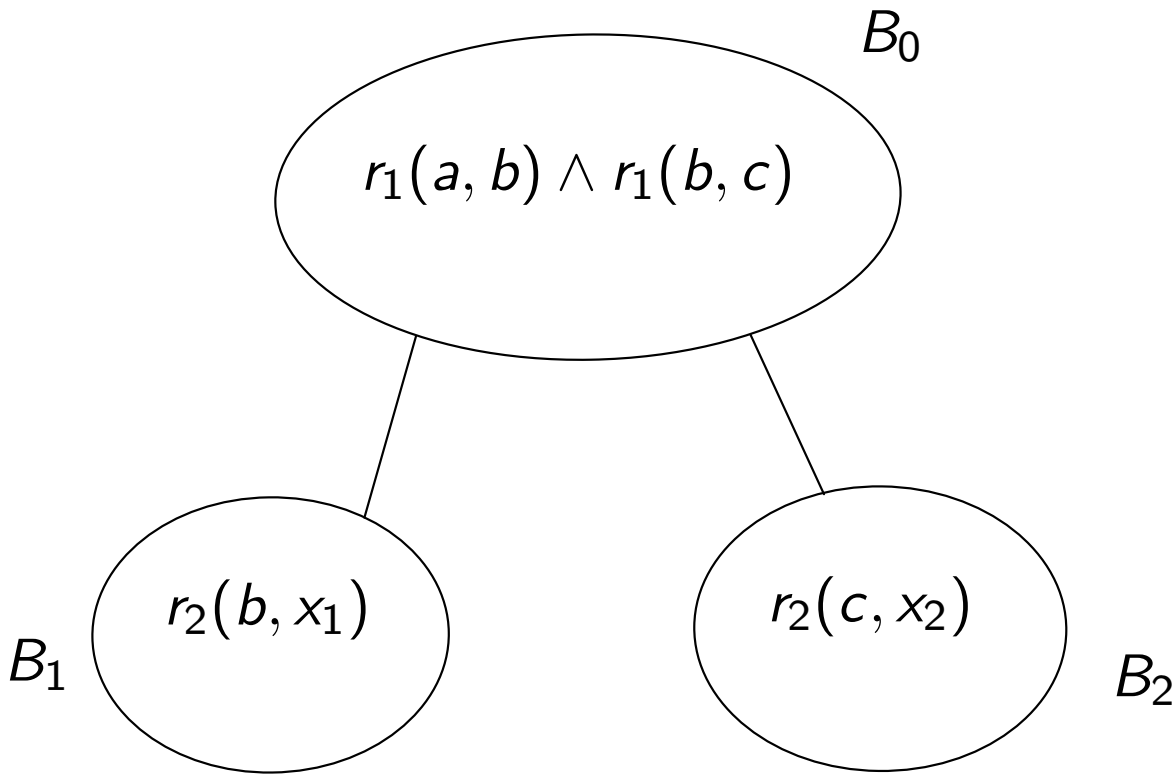
$$R_1 = r_1(x, y) \rightarrow r_2(y, z)$$
$$R_2 = r_2(x, y) \wedge r_1(x, z) \wedge r_2(z, t) \rightarrow r_3(y, t)$$

Un exemple de règles qui est *bts* mais pas *gbts*



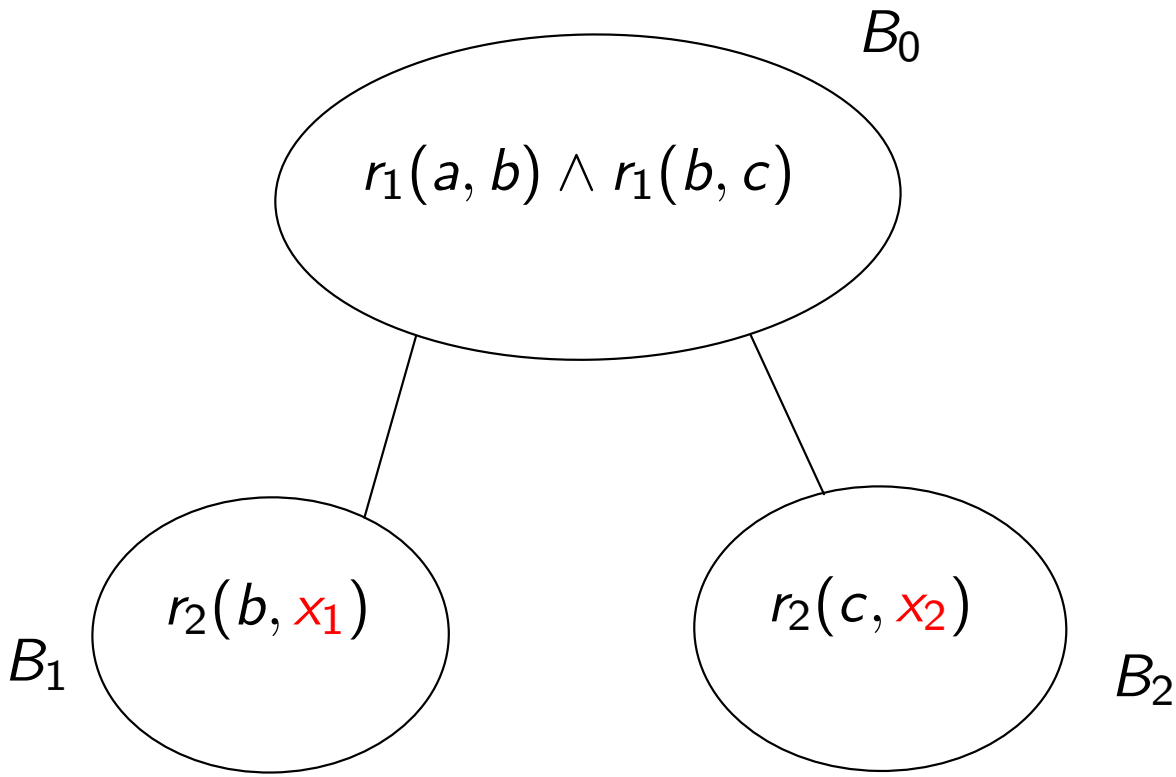
$R_1 = r_1(x, y) \rightarrow r_2(y, z)$
$R_2 = r_2(x, y) \wedge r_1(x, z) \wedge r_2(z, t) \rightarrow r_3(y, t)$

Un exemple de règles qui est *bts* mais pas *gbts*



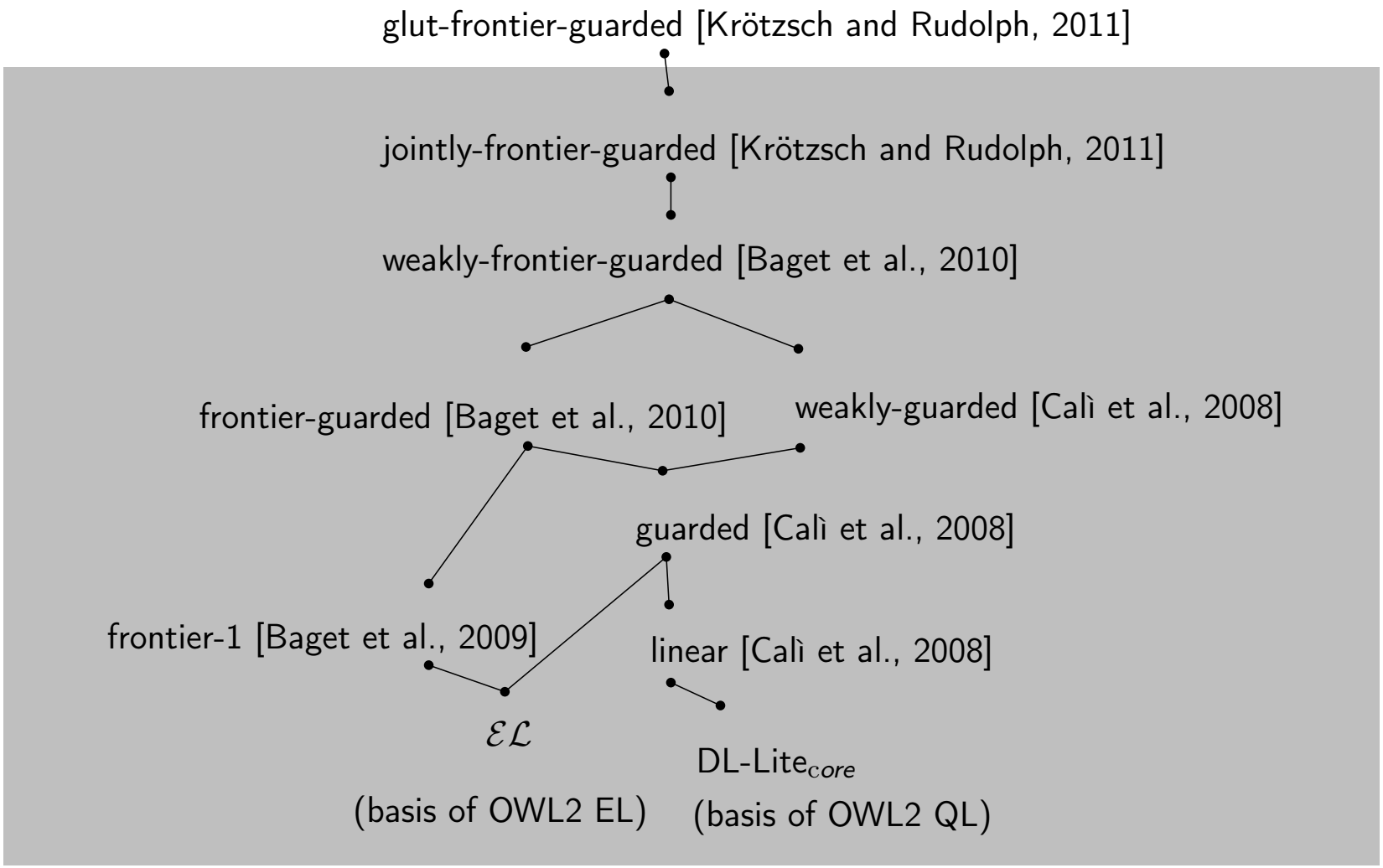
$R_1 = r_1(x, y) \rightarrow r_2(y, z)$
$R_2 = r_2(x, y) \wedge r_1(x, z) \wedge r_2(z, t) \rightarrow r_3(y, t)$

Un exemple de règles qui est *bts* mais pas *gbts*



$R_1 = r_1(x, y) \rightarrow r_2(y, z)$
$R_2 = r_2(x, y) \wedge r_1(x, z) \wedge r_2(z, t) \rightarrow r_3(y, t)$

gbts Classes



frontier-1 : la frontière est de taille 1

guarded : un atome du corps de la règle contient toutes ses variables

Idées principales de l'algorithme pour *gbts*

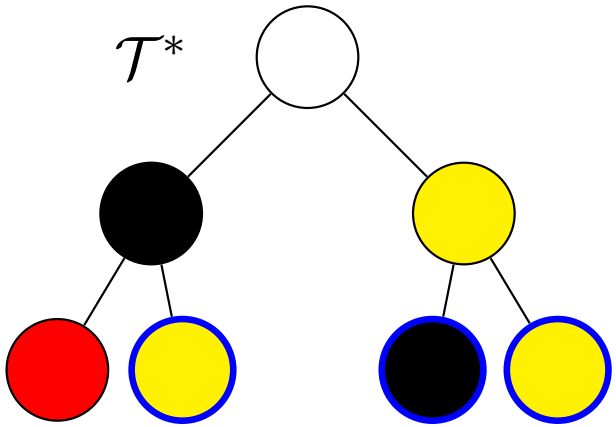
Représenter le modèle canonique de manière finie.

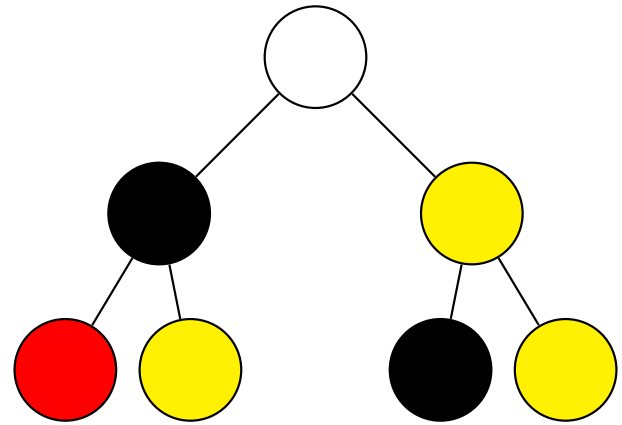
Principaux outils et étapes :

- définir une relation d'équivalence convenable sur les sacs de la décomposition arborescente du modèle canonique (*patterns* and *patterns abstraits*) ;
- calculer cette relation d'équivalence sans construire cette décomposition arborescente (*evolution* and *creation rules*) ;
- construire la représentation elle-même.

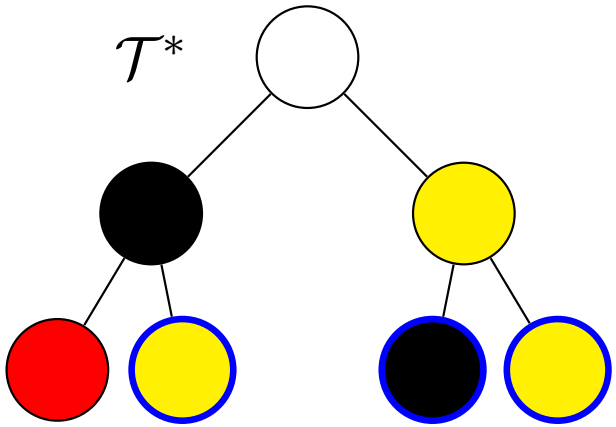
Outline

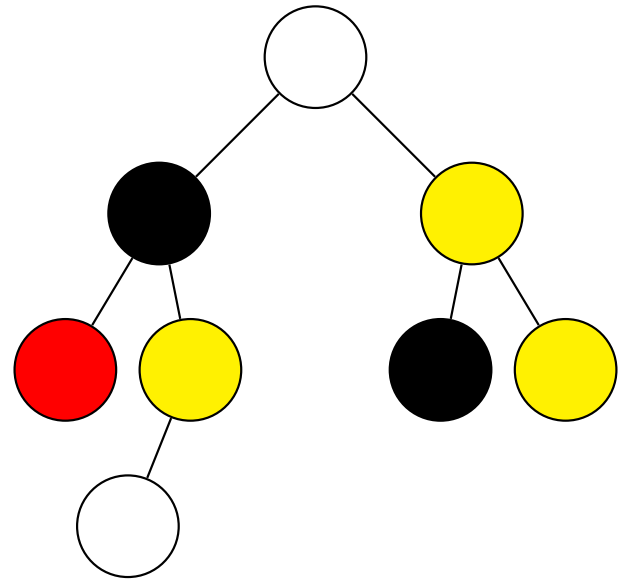
Représentation finie de la décomposition arborescente



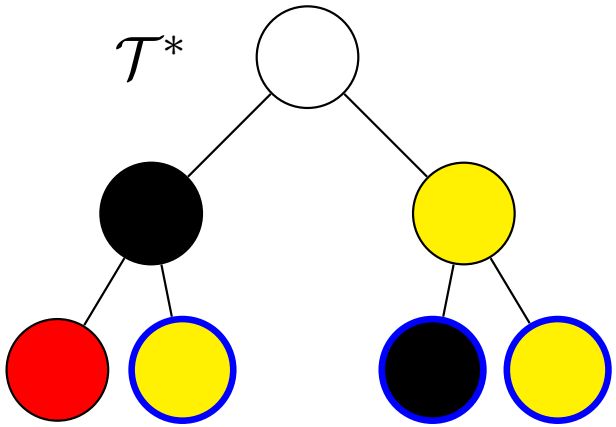


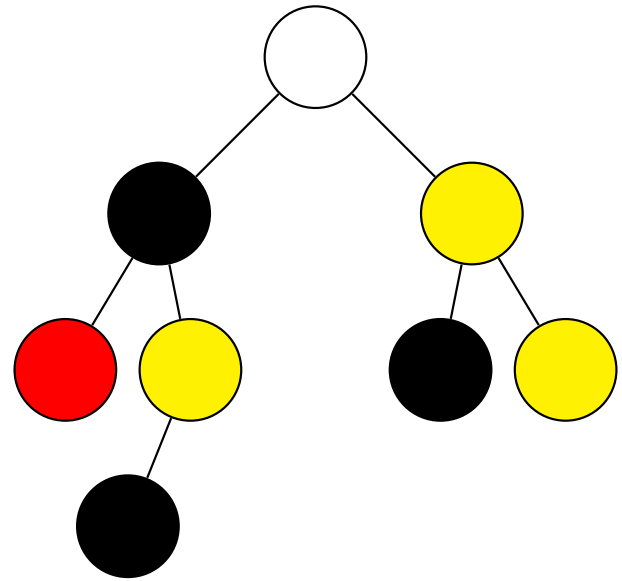
Représentation finie de la décomposition arborescente



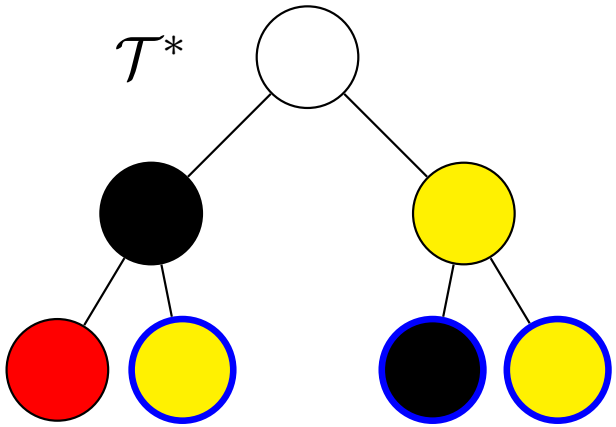


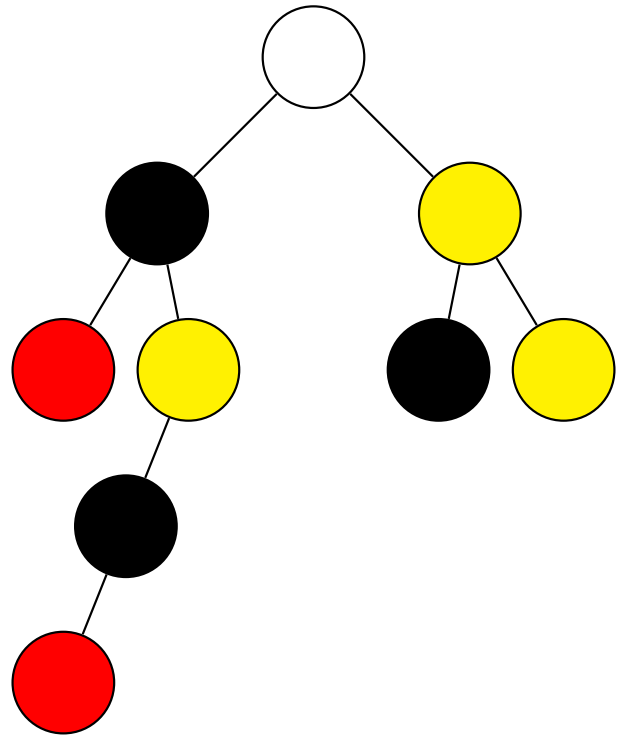
Représentation finie de la décomposition arborescente



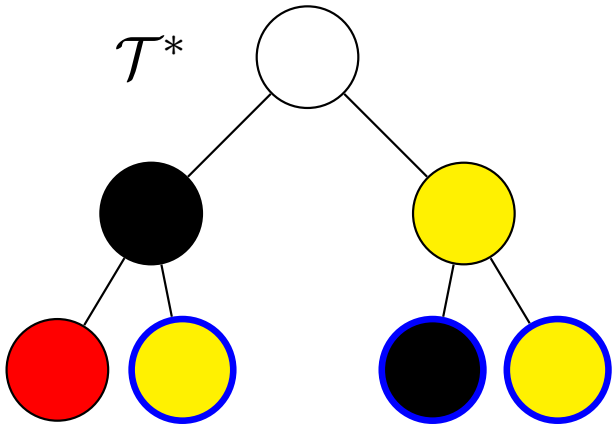


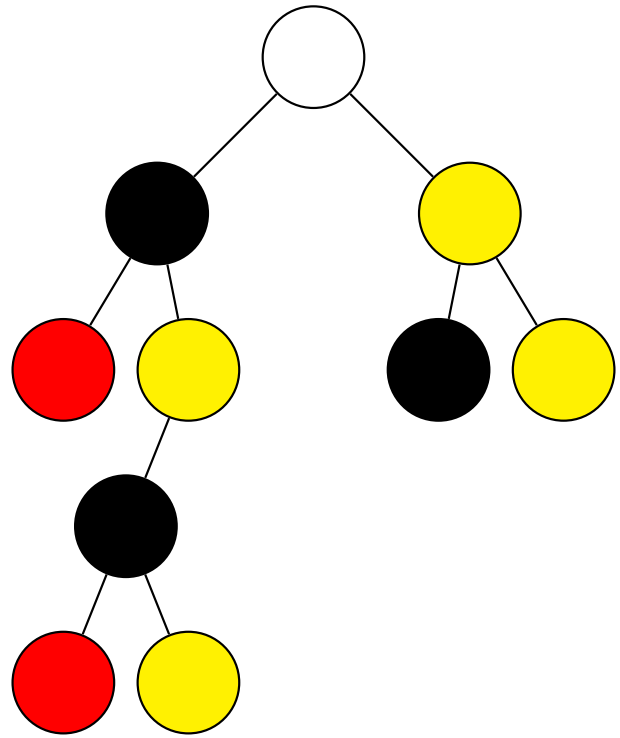
Représentation finie de la décomposition arborescente



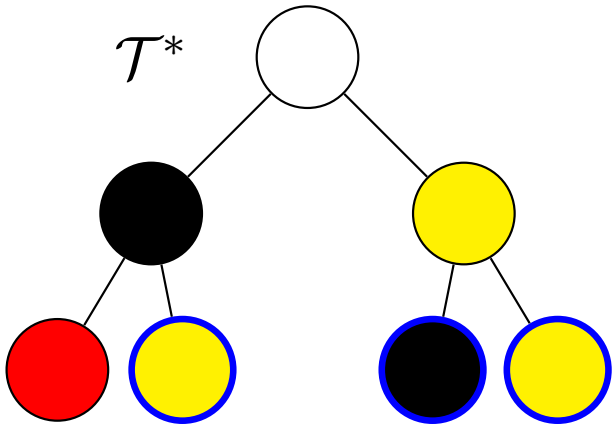


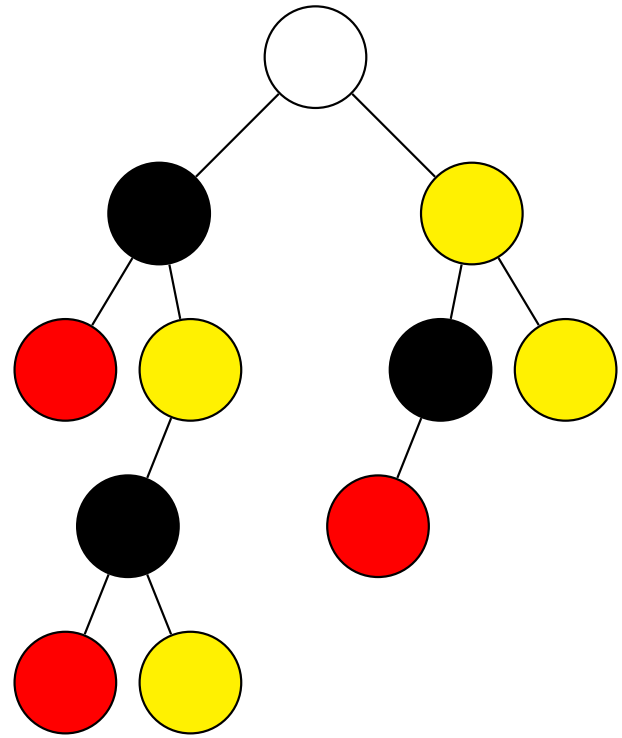
Représentation finie de la décomposition arborescente





Représentation finie de la décomposition arborescente





...

Patterns et patterns abstraits

- relation d'équivalence se sert de *patterns*
- les sacs d'un arbre de dérivation sont étiquetés par des patterns
- intuitivement : cela représente comment les corps de règles sont envoyés dans ce sac

Equivalence

Deux sacs avec même pattern abstrait sont à la racine d'arbres isomorphes dans la décomposition arborescente du modèle canonique

Calcul des patterns abstraits

- **but** : calcul des patterns abstraits
- **outil** : règles d'évolution et de création

Complexité

Classe	**Combinée**	**Données**
gbts	2ExpTime-c [T. et al. 12]	ExpTime-c [Baget et al. 11]
j-fg	2ExpTime-c [Krötzsch et al. 11]	ExpTime-c [Krötzsch et al. 11]
wfg	2ExpTime-c [Baget et al. 11]	ExpTime-c [Baget et al. 11]
fg	2ExpTime-c [Baget et al. 11]	PTime-c [Baget et al. 11]
fr1	2ExpTime-c [Baget et al. 11]	PTime-c [Baget et al. 11]
wg	2ExpTime-c [Calì et al. 08]	ExpTime-c [Calì et al. 08]
guarded	2ExpTime-c [Calì et al. 08]	PTime-c [Calì et al. 08]
linear	PSpace-c [Calì et al. 10]	in AC_0 [Calì et al. 09]

Résumé

- définition d'une nouvelle classe décidable et expressive pour OCQA [Baget, Mugnier, Rudolph, T., IJCAI'11]
- construction d'un algorithme optimal dans le pire des cas pour cette classe

  [T., Baget, Mugnier, Rudolph, KR'12]
- adaptation de cet algorithme à des sous-classes [T., Baget, Mugnier, Rudolph, KR'12] et [T., DL'12]
- design, implementation d'un prototype and et première évaluation d'un algorithme de réécriture [T., IJCAI'13]

Travaux futurs

- implémentation de l'algorithme pour *gbts*
- extension de l'approche
- réécriture au delà des formules du premier-ordre

Remerciements

- à Jean-François et Marie-Laure pour leur encadrement au top ;
- à GraphIK pour l'environnement ;
- au jury (Georg Gottlob, Carsten Lutz, Christophe Paul, Marie-Christine Rousset) ;
- au jury du prix de thèse AFIA ;
- à la DAAD et à la JSPS ;
- et à vous, pour votre attention !