

Règles avec négation: des bonnes dérivations aux modèles stables

Cours confiné GraphIK

14/04/2020

Application des règles avec
négation

Application des règles existentielles

$B \rightarrow H$



F

h : homomorphisme de B dans H = *déclencheur* de la règle

Règle *applicable* = il existe un déclencheur

$$\alpha(F, R, h) = F \cup h^s(H)$$

Application des règles avec négation

$$B^+, \text{ not } B_1^-, \dots, \text{ not } B_i^-, \dots, \text{ not } B_k^- \rightarrow H$$

h = déclencheur
de la règle

F

h' homomorphisme de B_i^- dans F qui étend h
= h est bloqué par h'

Règle *applicable* = il existe un déclencheur *non bloqué*

Négation par l'échec!

not $\neq \neg$

$$\alpha(F, R, h) = F \cup h^s(H)$$


Restriction: donner un sens à l'application

$p(X), \text{ not } q(X, Y) \rightarrow r(Y)$



$p(a)$



$\left\{ \begin{array}{l} r(a) \\ r(b) \end{array} \right.$ 

Même si b est une constante qui n'apparaît pas dans F ?
Problème: plus d'ajout de données « dans le voisinage »

Toute variable de la tête apparaît soit dans le corps positif, soit:

- *n'apparaît dans aucun corps négatif (c'est donc une variable exist.)*
- *N'est pas dans la portée des variables du corps négatif (renommage auto.)*

Exemple d'application

$p(X)$, not $q(X, Y)$, $p(Y)$, not $q(Y, X) \rightarrow r(X)$

$F \left\{ \begin{array}{l} p(a) \\ q(a, b) \end{array} \right.$



$r(a) ?$

$G \left\{ \begin{array}{l} p(a) \\ q(b, a) \end{array} \right.$



$r(a) ?$

Bonnes dérivations

Dérivation avec des règles existentielles

\mathcal{R}

$$F_{i+1} = \alpha(F_i, R, h)$$

F

$$F = F_0, F_1, F_2, \dots, F_i, F_{i+1}, \dots, F_k, \dots$$

Résultat de la dérivation

Dérivation complète

$$F^* = \bigcup F_i$$

Tout ce qui est applicable sur le résultat a déjà été appliqué



Plus compliqué quand on effectue des simplifications non monotones...

Toutes les dérivations complètes ont un résultat équivalent.
Ce résultat est un *modèle universel*.

Dérivation des règles avec négation

\mathcal{R}

$$F_{i+1} = \alpha(F_i, R, h)$$

F

$$= F_0, F_1, F_2, \dots, \overbrace{F_i, F_{i+1}, \dots, F_k}^{\text{Dérivation complète}}, \dots$$

Résultat de la dérivation

Dérivation complète

$$F^* = \bigcup F_i$$

Tout ce qui est applicable sur le résultat a déjà été appliqué

Que peut-on dire de ces dérivations ?

Exemple introductif et néanmoins significatif

$$(R_1) \quad p(X), \text{ not } q(X) \rightarrow r(X)$$

$$(R_2) \quad p(X) \rightarrow q(X)$$

$p(a)$

$$p(a) \xrightarrow[\text{q(a) absent}]{\text{application } R_1,} p(a), r(a) \xrightarrow[\text{q(a) apparaît}]{\text{application } R_2,} p(a), r(a), q(a)$$

On voudrait des applications *persistantes* dans la dérivation: toute application devrait rester applicable.

$$p(a) \xrightarrow[\text{q(a) absent}]{\text{application } R_1,} p(a), r(a) \quad \begin{array}{l} R_2 \text{ est applicable,} \\ \text{mais on ferme les yeux} \end{array}$$

On voudrait des dérivations complètes.

$$p(a) \xrightarrow{\text{application } R_2} p(a), r(a) \quad \begin{array}{l} R_1 \text{ n'est plus applicable,} \\ \text{on a bien fini} \end{array}$$

Bonne dérivation : persistante et complète.

Bonnes dérivations: existence ?

$p(X), \text{ not } q(X) \rightarrow q(X)$

$p(a)$

$p(a)$ R est applicable,
mais on ferme les yeux

Dérivation non complète

application R,
q(a) absent
 $p(a) \longrightarrow p(a), q(a)$ R n'est plus applicable,
après son application

Dérivation non persistante

Pas de bonne dérivation

Exercice: une implémentation de \perp

Notre programme contient la règle:

$\perp(X), \text{ not } q(X) \rightarrow q(X)$

On veut qu'*aucune bonne dérivation ne puisse contenir \perp*

- Ca marche?
- Ca ne marche pas? Patchable?

Bonnes dérivations: unicité ?

$$p(X), \text{ not } q(X) \rightarrow r(X)$$

$$p(X), \text{ not } r(X) \rightarrow q(X)$$

$$p(a)$$

application R d'une règle

$$p(a) \longrightarrow p(a), r(a) \quad \text{L'autre n'est plus applicable}$$

application R d'une règle

$$p(a) \longrightarrow p(a), q(a) \quad \text{L'autre n'est plus applicable}$$

Exercice: une implémentation de \vee

Notre programme contient les règles:

$$q \vee r(X), \text{ not } q(X) \rightarrow r(X)$$

$$q \vee r(X), \text{ not } r(X) \rightarrow q(X)$$

On veut que *toute bonne dérivation contenant $q \vee r$ contient également q ou r* .

- Ca marche?
- Ca ne marche pas?

Est-ce un ou exclusif? Sinon, comment le modéliser?

Bonne dérivation

Bonne dérivation

Résultats
incomparables

Bonnes dérivations: 0, 1, 2, ..., beaucoup ?

$p(X)$, not $q_1(X) \rightarrow r_1(X)$

$p(X)$, not $r_1(X) \rightarrow q_1(X)$

$p(X)$, not $q_2(X) \rightarrow r_2(X)$

$p(X)$, not $r_2(X) \rightarrow q_2(X)$

\vdots

$p(X)$, not $q_n(X) \rightarrow r_n(X)$

$p(X)$, not $r_n(X) \rightarrow q_n(X)$

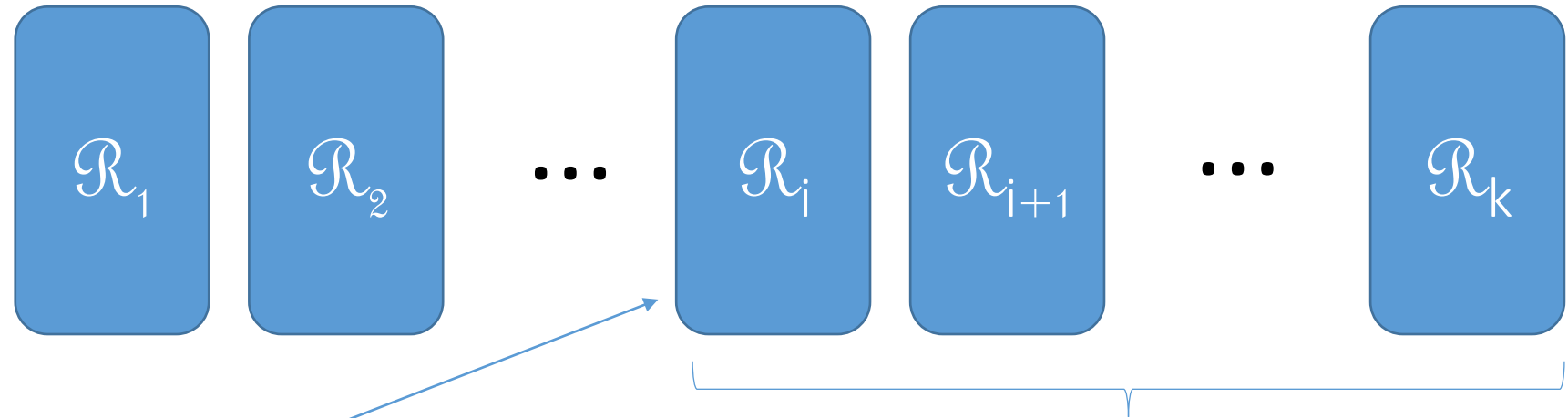
$p(a)$

2^n résultats de bonnes dérivations distincts

Bonnes dérivations et Datalog stratifié

Partition ordonnée des règles en *strates*

Règles stratifiables



Outil « faible »:
graphe de
dépendance
des prédicats

Si un ensemble d'atomes peut bloquer
l'application d'une règle de cette strate

Aucune règle des strates supérieures ne pourra
participer à la génération de cet ensemble d'atomes.

Une *dérivation stratifiée* respecte l'ordre d'une telle stratification (possible car Datalog f.e.s.)
Toute dérivation stratifiée complète donne le même résultat.

- Toute dérivation stratifiée est persistante.
- Toute dérivation persistante de règles stratifiables peut se réordonner en une dérivation stratifiée.



Si les règles sont stratifiables, toute bonne
dérivation donne le même résultat.

Des bonnes dérivations aux
modèles stables

Intermède ludique: transformation en propositionnel

Phase 1: Skolemisation

$$p(A, b, C) \longrightarrow p(a, b, c)$$

« freeze » de F

$$p(X, Y, Z), \text{ not } q(X, T) \longrightarrow r(X, U), r(Y, V)$$



$$p(X, Y, Z), \text{ not } q(X, T) \longrightarrow$$

$$r(X, f_U(X, Y, Z)), r(Y, f_V(X, Y, Z)) \text{ corps}$$

$$r(X, f_U(X, Y)), r(Y, f_V(X, Y)) \text{ frontière}$$

$$r(X, f_U(X)), r(Y, f_V(Y)) \text{ pièces}$$

Les résultats d'une bonne (oblivious) dérivation d'une KB ou de sa corps-skolémisation sont « équivalents » : toute BCQ (sans symbole fonctionnel) déductible de l'un est déductible de l'autre.

Voir pourquoi les autres skolemisations peuvent donner des résultats différents dans (Baget et al., NMR 2014)

Intermède ludique: transformation en propositionnel

Phase 2: Normalisation

Une règle avec négation est *normale* quand toutes les variables des corps négatifs apparaissent dans le corps positif.

$$p(X, Y, Z), \text{ not } q(X, T) \rightarrow r(X, f_U(X, Y, Z)), r(Y, f_V(X, Y, Z))$$



$$\begin{aligned} & p(X, Y, Z), \text{ not } C_1(X) \rightarrow r(X, f_U(X, Y, Z)), r(Y, f_V(X, Y, Z)) \\ & Q(X, T) \rightarrow C_1(X) \end{aligned}$$

Les résultats d'une bonne dérivation d'une KB ou de sa normalisation sont « équivalents » : toute BCQ (sans nouveau prédicat) déductible de l'un est déductible de l'autre.

Voir plus tard l'utilité de cette normalisation dans le « grounding »: on comprendra pourquoi les concepteurs d'ASP, qui utilisent ce grounding, imposent des règles normales.

Intermède ludique: transformation en propositionnel

Phase 3: Instanciation

(d'une KB skolemisée, normale, sans variable existentielle)

$p(X), \text{not } q(X) \rightarrow r(X)$

$s(X, Y) \rightarrow q(X)$

$p(a), q(b).$

Remarque: l'instanciation génère H^k règles par règle initiale.

- H : taille du domaine de Herbrand
- k : nombre de variables de la règle

3.1: Calcul du domaine de Herbrand

$\mathcal{H} = \{a, b\}$

3.1: Instanciation de chaque règle

$p(a), \text{not } q(a) \rightarrow r(a)$

$p(b), \text{not } q(b) \rightarrow r(b)$

$s(a, a) \rightarrow q(a)$

$s(a, b) \rightarrow q(a)$

$s(b, a) \rightarrow q(b)$

$s(b, b) \rightarrow q(b)$

$p(a), q(b).$

Les résultats d'une bonne dérivation d'une KB skolemisée normale sans variable existentielle ou de son instanciation sont identiques.

Intermède ludique: transformation en propositionnel

Phase 3: Instanciation

(d'une KB skolemisée, normale,
cas des variables existentielles)

$p(X), \text{not } q(X) \rightarrow r(X, f(X))$

$s(X, Y) \rightarrow q(X)$

$p(a), q(b).$

3.1: Calcul du domaine de Herbrand

$\mathcal{H} = \{a, b, f(a), f(b), f(f(a)), f(f(b)), \dots\}$

3.1: Instanciation de chaque règle

$p(a), \text{not } q(a) \rightarrow r(a)$

$p(b), \text{not } q(b) \rightarrow r(b)$

$p(f(a)), \text{not } q(f(a)) \rightarrow r(f(a))$

...

Les résultats d'une bonne dérivation d'une
KB skolemisée normale ou de son instanciation
sont identiques.

MAIS l'ensemble de règles est infini...

Intermède ludique: transformation en propositionnel

Pourquoi la normalisation?

1. Dérivation de la KB initiale

$p(a), s(a, b).$

2. Dérivation de la KB instanciée

$p(a), s(a, b). \longrightarrow p(a), s(a, b), r(a).$

même si 2 est bloquée,
1 ne l'est pas.

3. Dérivation de la KB normalisée instanciée

$p(a), s(a, b). \rightarrow p(a), s(a, b), q(a).$

application de 4

L'application de 1 est bloquée.

$p(X), \text{not } s(X, Y) \rightarrow r(X)$

$p(a), s(a, b).$

$p(a), \text{not } s(a, a) \rightarrow r(a) \quad 1$

$p(a), \text{not } s(a, b) \rightarrow r(a) \quad 2$

$p(b), \text{not } s(b, a) \rightarrow r(b) \quad 3$

$p(b), \text{not } s(b, b) \rightarrow r(b) \quad 4$

$p(a), s(a, b).$

$p(a), \text{not } q(a) \rightarrow r(a) \quad 1$

$p(b), \text{not } q(b) \rightarrow r(a) \quad 2$

$s(a, a) \rightarrow q(a) \quad 3$

$s(a, b) \rightarrow q(a) \quad 4$

$s(b, a) \rightarrow q(b) \quad 5$

$s(b, b) \rightarrow q(b) \quad 6$

$p(a), s(a, b).$

Intermède ludique: transformation en propositionnel

Une KB instanciée est une KB propositionnelle (éventuellement infinie).

Codage

| p(a) | p(b) | s(a, a) | s(a, b) | s(b, a) | s(b, b) | r(a) | r(b) |
|------|------|---------|---------|---------|---------|------|------|
| a | b | c | d | e | f | g | h |

$p(a), \text{not } s(a, a) \rightarrow r(a)$
 $p(a), \text{not } s(a, b) \rightarrow r(a)$
 $p(b), \text{not } s(b, a) \rightarrow r(b)$
 $p(b), \text{not } s(b, b) \rightarrow r(b)$
 $p(a), s(a, b).$



$a, \text{not } c \rightarrow g$
 $a, \text{not } d \rightarrow g$
 $b, \text{not } e \rightarrow h$
 $b, \text{not } f \rightarrow h$
 $a, d.$

Les résultats d'une bonne dérivation d'une KB instanciée et de son codage sont identiques (au codage près).

Conclusion de l'intermède (pas si ludique que ça): on peut transformer notre KB (règles avec négation) en une KB propositionnelle (avec négation par l'échec) qui « conserve » les résultats des bonnes dérivations.

Bonnes dérivations: version propositionnelle

Supposons E (un ensemble d'atome) résultat d'une bonne dérivation d'une KB propositionnelle K .

$$F_0, F_1, F_2, \dots, F_i, F_{i+1}, \dots, F_k, \dots \quad E = \bigcup F_i$$

Considérons une règle quelconque de K .

$$B^+, \text{ not } B_1^-, \dots, \text{ not } B_i^-, \dots, \text{ not } B_k^- \rightarrow H$$

Si cette règle n'a pas été appliquée dans cette dérivation complète, c'est soit *qu'elle n'est pas déclenchable* ($B^+ \not\subseteq E$), soit qu'elle a été bloquée (il existe $B_i^- \subseteq E$). Supprimer cette règle de K ne change pas cette dérivation.

Si cette règle a été appliquée dans cette dérivation persistante, elle n'est bloquée à aucune étape de la dérivation (il n'existe pas $B_i^- \subseteq E$). Remplacer cette règle par $B^+ \rightarrow H$ ne change pas cette dérivation.

On note $K|_E$ la base de connaissances ainsi *réduite* par E .
On a bien $K|_E^* = E$. La réciproque est vraie!

Bonnes dérivations: premier algorithme

1. Propositionalisation



Cette étape part à l'infini si domaine de Herbrand infini (i.e. si vars exist., sauf certaines optimisations)

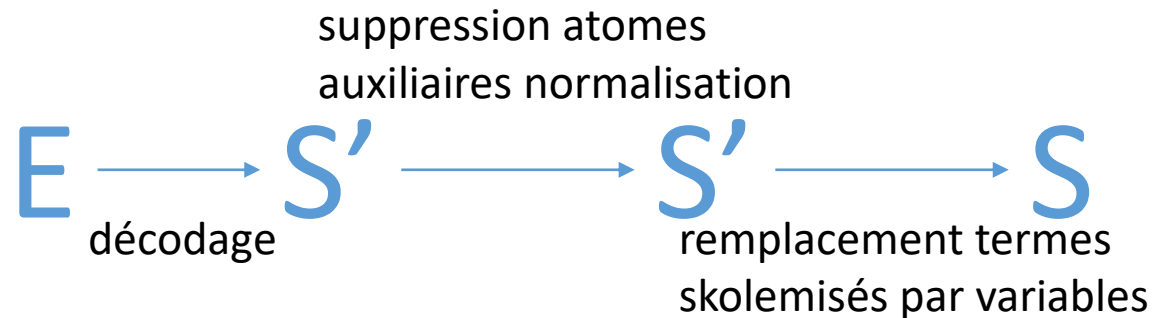
2. Générer

On devine un sous-ensemble E des atomes propositionnels

3. Tester

On teste si $E = P|_E^*$. Par définition (fin du suspense): E est un *modèle stable* de P .

4. Transformation inverse



S est le résultat d'une bonne dérivation de K ssi E est un modèle stable de P .

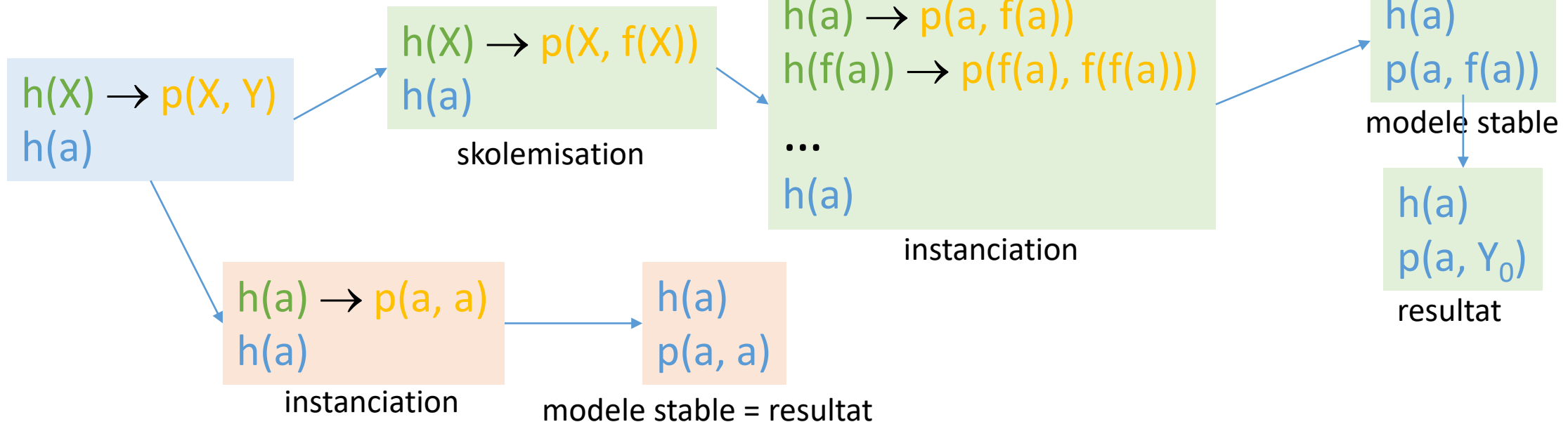
Différences avec ASP (Answer Set Programming)

1. Propositionalisation (i.e. *grounding*)



Pas de normalisation: les règles doivent déjà être sous forme normale (différence syntaxique)

Pas de skolemisation: (différence sémantique)



La skolemisation préserve la sémantique des règles existentielles (« tous les humains ont un parent »), ASP considère des variables universelles en tête (« tout est parent d'un humain »). L'instanciation reste finie, et on peut « forcer » les variables existentielles en écrivant directement la forme skolemisée (mais on perd l'instanciation finie). Identiques si Datalog.

Conclusion (enfin, l'heure doit être dépassée)

Extension des règles existentielles à la négation par l'échec basée sur la sémantique (propositionnelle) des modèles stables (via *une* skolemisation)

Si pas de variable existentielle, même sémantique que ASP

Sinon, sémantique différente afin d'être une vraie extension des règles existentielles (notre algorithme, en supposant qu'il puisse calculer à l'infini, nous retourne un seul modèle stable qui est le modèle universel dans le cas des KB positives).

Dans les prochains cours:

Un algorithme permettant de s'affranchir de l'instanciation, et remplaçant le generer/tester par un BT (ASPERIX)

Des exemples de modélisation utilisant ces règles avec négation

Simple: résolution de k-coloration

Moyen: le loup, la chèvre, le chou, voire le Wumpus

Compliqué: évolution d'un système complexe

Expert (voir avec Madalina): architecture agents rationnels/environnement