RÈGLES EXISTENTIELLES: CHASE

THÉORIE DES BASES DE CONNAISSANCES
HMIN312M
COURS DE ML MUGNIER

EXISTENTIAL RULES

$\forall X \, \forall Y \, (\underbrace{\text{Body } [X,Y]} \rightarrow \exists Z \, \underbrace{\text{Head } [X,Z]})$

X, Y, Z : *sets of variables*

any **positive conjunction** (without functional symbols)

$\forall x \, (\text{actor}(x) \rightarrow \exists z \, \text{play}(x,z))$

$\forall x \, \forall y \, (\text{siblingOf}(x,y) \rightarrow \exists z \, (\text{parentOf}(z,x) \wedge \text{parentOf}(z,y)))$

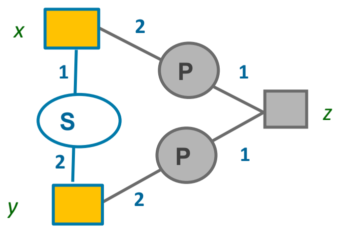
Key point: ability to assert **the existence of unknown entities**

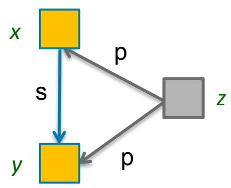
Crucial for representing ontological knowledge in **open domains**

Graph View of (Existential) Rules

$\forall X\ \forall Y\ (\underbrace{\text{Body } [X,Y]}_{\text{graph}} \rightarrow \exists Z\ \underbrace{\text{Head } [X,Z]}_{\text{graph}})$

∀x ∀y (siblingOf(x,y) → ∃ z (parentOf(z,x) ∧ parentOf(z,y)))





The rule head has 2 kinds of variables (or unlabelled term nodes):
- **frontier**: shared with the body
- **existential**

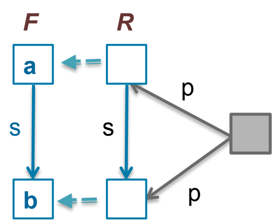
Generation of fresh (unknown) individuals

R = ∀x ∀y (siblingOf(x,y) → ∃ z (parentOf(z,x) ∧ parentOf(z,y)))

F = siblingOf(a,b)

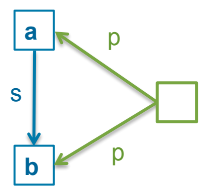
R is **applicable** to *F* if there is a **homomorphism** *h* from *body(R)* to *F*

x → a
y → b



Applying *R* to *F* w.r.t. *h* produces *F* ∪ *h(head(R))*
where a **fresh** *variable (a «* null *») is created for each* existential *variable in R*

F' = **∃ z0** (siblingOf(a,b) ∧ parentOf(z0,a) ∧ parentOf(z0,b))



Notation (when needed) : **α(F,R,h) = F ∪ h^{safe}(head(R))**

where h^{safe} is a substitution of var(head(R))
such that: h^{safe}(x) = h(x) if x in frontier(R)
otherwise h^{safe}(x) is a fresh variable (a null)

Knowledge bases with existential rules

$\mathcal{K} = (F, \mathcal{R})$ where

$\mathcal{R}$ is a set of **existential rules**

F is a set of **facts** (rules with an empty body): existential conjunctions of atoms

Forward chaining called « **chase** » (let us still denote by *F** the result of the chase)

Main change with respect to Datalog rules: *F** can be **infinite**

R = person(x) → ∃ y hasParent(x,y) ∧ person(y)

F = person(a)

∧ person(y0) ∧ hasParent(a, y0)

∧ person(y1) ∧ hasParent(y0, y1) Etc.

but it remains a **universal model**

Hence, for BCQs : **K ⊨ *q* iff *q* ≥ *F***

Other changes: **F* is not unique** (but all F*we will see are logically equivalent)

Derivation

- **Trigger** for a set of facts F: **(R,h)** | h homomorphism from body(R) to F
- **Derivation**: $(F_0 = F)\ (R_1,h_1)\ F_1\ (R_2,h_2)\ F_2$, ...

  where for all i, (h_i, R_i) trigger for F_{i-1}

  and $F_i = \alpha(F_{i-1},R_i,h_i)$

  When the triggers are not needed, we note $(F_0=F), F_1, F_2$, ...
- The notion of derivation can be generalized by

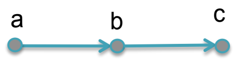
  taking F_i equivalent to $\alpha(F_{i-1},R_i,h_i)$ instead of equal
- **Different chase variants** with their own rule application criteria

→ notion of *legal* rule application (or: *active* trigger (R_i, h_i)):
A chase variant considers only derivations with active triggers

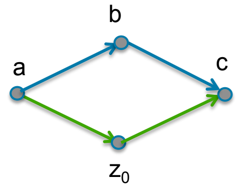
DIFFERENT VARIANTS OF THE CHASE

All variants we will see compute **universal models** of the KB
but they differ on how they handle **redundancies** possibly caused by nulls

p(a,b), p(b,c)



p(a,b), p(b,c),
∃ z_0 p(a,z_0) p(z_0,c)



$z_0 \mapsto b$

Core: set of atoms without homomorphism to one of its strict subsets

OBLIVIOUS CHASE

Oblivious (or **naive**): « performs all rule applications according to all new triggers »

A trigger (R,h) to F_i is *active on F_i* iff this trigger has *not* already been used in the derivation from F_0 to F_{i-1}

R = p(x,y) → ∃ z p(x,z)

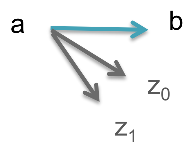
stupid rules to keep examples simple!

F = p(a,b)

p(a,z_0)

p(a,z_1)

...



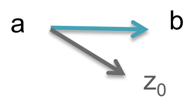
infinite derivation

OBLIVIOUS / SEMI-OBLIVIOUS = SKOLEM CHASE

Semi-oblivious: consider only homomorphisms that differ on the rule frontier (x)

A trigger (R,h) to F_i is *active on F_i* iff there is *no* trigger (R,h')
such that *h'(x) = h(x)* for all *x* in frontier(R)
in the derivation from F_0 to F_{i-1}

F = p(a,b)   *R* = p(x,y) → ∃ z p(x,z)



Skolem chase: similar behavior

(1) skolemize rules: in R, replace each existential variable z by a function f_R^z(frontier(R))

(2) perform the oblivious chase on skolemized rules

R = p(x,y) → p(x,f(x))

p(a,b)   p(a,f(a))

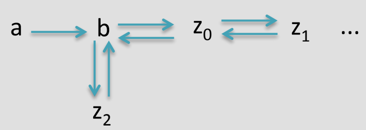
Skolemization can be seen as a way of naming existential variables and « tracking » the nulls created during the semi-oblivious chase

RESTRICTED (AKA STANDARD) CHASE

Restricted: do not perform a rule application that brings *only* redundant information

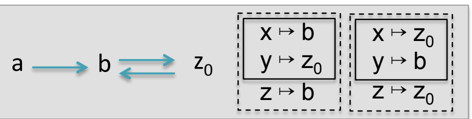
A trigger (R,h) to F_i is *active on F_i* iff
h cannot be extended to homomorphism *h'*: body ∪ head → F_i

F : p(a,b)   *R* : p(x,y) → ∃ z p(y,z), p(z,y)



(semi-) oblivious chase:

infinite



restricted chase:

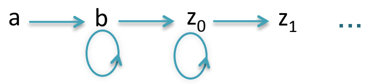
halts after one rule application

RESTRICTED CHASE: NATURAL BUT TRICKY

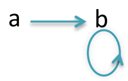
- For the same KB, some derivations may halt while others may not

F : p(a,b)   *R*$_1$: p(x,y) → ∃ z p(y,z)
R$_2$: p(x,y) → p(y,y)

If R$_1$ is always applied before R$_2$ for a given homomorphism of p(x,y):



If R$_2$ is applied first:

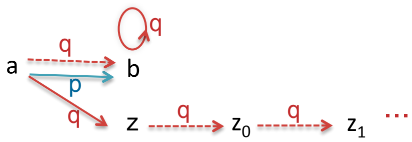


CORE CHASE

Iterate:
(1) perform a finite number of rule applications as in the restricted chase
(2) compute the core of the result

F : p(a,b), q(b,b), q(a,z)   where z is a variable

R$_1$: p(x,y) → q(x,y)
R$_2$: q(x,y) → ∃ z q(y,z)



The restricted chase only checks redundancy of **newly** added atoms ⇒ infinite here

The core chase outputs { p(a,b), q(b,b), q(a,b) }

The core chase allows to detect **global** redundancies

WHEN DOES A CHASE HALT?

- **Terminating** derivation:
  (1) finite and (2) there is no active trigger on the last factbase
- A chase derivation has to be **fair**: no active trigger is indefinitely delayed
  Formally: if (R,h) is an active trigger on F_i
  then there is F_j with $j > i$ such that F_j is obtained by applying (R,h)
  or (R,h) is not active anymore on F_j

Terminating = finite and fair

R_1: $p(x,y) \rightarrow \exists z\ p(y,z)$
R_2: $p(x,y) \rightarrow p(y,y)$

F= p(a,b)

unfair infinite derivation: apply only R_1 ...

(semi-) oblivious: all fair derivations are infinite

restricted: some terminating derivations,
some infinite fair derivations

core: all fair derivations are terminating

For a chase variant C, C **halts** on a KB K if all **fair** derivations on K are finite

IN SHORT

All previous chase variants compute **universal** models of a KB

They can be **strictly ordered wrt termination**:

oblivious < semi-oblivious = skolem < restricted < core

[X < Y means that: for any KB K, if X-chase halts on K then Y-chase halts on K
and there is a KB on which Y-chase halts but not X-chase]

Only the **core** chase halts if and only if the KB admits a **finite** universal model
but it is costly (involves homomorphisms from the whole factbase)

The **O**, **S-O** and **core** chases yield a **unique** result (up to the name of nulls):
all fair derivations for a given chase variant yield the same result on a given KB
but not the **R chase**: we can even have finite and infinite fair derivations

The **R chase** seems to achieve a good tradeoff
redundancy elimination / efficiency of computation (when it stops)
but its behavior is difficult to control

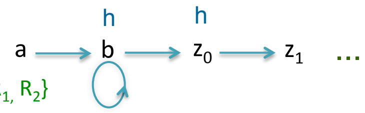
TRICKY RESTRICTED CHASE

Open question:
is there an ordering strategy that terminates more often than the others?

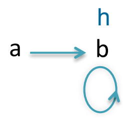
- **Breadth-first ordering** is a natural candidate (iterate:
  (1) compute all rule body homomorphisms to the current factbase,
  (2) apply all active triggers according to these homomorphisms)
- however, it is **not optimal for restricted chase** termination

R_1: $p(x,y) \rightarrow \exists z\ p(y,z)$
R_2: $p(x,y) \rightarrow h(y)$
R_3: $h(x) \rightarrow p(x,x)$

$F = p(a,b)$
$p(b,z_0), h(b)$ $\{R_1, R_2\}$
$p(z_0,z_1), h(z_0), p(b,b)$ $\{R_1, R_2, R_3\}$



Optimal order: apply R_2 then R_3 (ie delay application of R_1)



- Usual heuristic: at each step, first saturate with all datalog rules, then apply an existential rule

→ would be optimal on this example, is it always the case?