Implementing the Core Chase for the Description Logic ALC

Maël Abily

June 27, 2021

1 Introduction

An important problem in database is the conjunctive query entailment. This problem can be described in a first order logic background: Given a knowledge base $O = (R, F)$ where F is a set of conjunctive formulas (that are formulas constructed only with conjunction and existential quantification) and R is a set of rules (that are, if $\vec{u}$ represents a tuple of variables, formulas of the form $\forall\vec{x}.\forall\vec{y}.(A(\vec{x},\vec{y}) \rightarrow \exists\vec{z}.B(\vec{x},\vec{z}))$), and given a query Q (that is a conjunctive formula), determine if the knowledge base O entails the query Q. We usually use reasoning algorithms to answer this problem. We can notice that in practice, a lot of formulas can be express via conjunctive formulas.

Example 1.1. For the knowledge base O composed of the set of conjunctive formulas $F = \{\mathit{Father}(\mathit{Michel})\}$ and the set of rules $R = \{\forall x.\mathit{Father}(x) \rightarrow \exists y.\mathit{IsTheSonOf}(y, x)\}$, and for the query $Q = \exists y.\mathit{IsTheSonOf}(y, \mathit{Michel})$, the knowledge base O entails Q.

By definition, a knowledge base O entails a query Q if every model of O is a model of Q. It is not practical because a knowledge base can have an infinite number of model.

To deal with this problem, we can compute an universal model of the knowledge base O. That is a model of O that is entailed by all the models of O. If a such model U exists, we just need to show that U entails Q to conclude that O entails Q. Hence, to solve query entailment, we just have to compute a finite universal model U for a given input knowledge base and look if U entails Q.

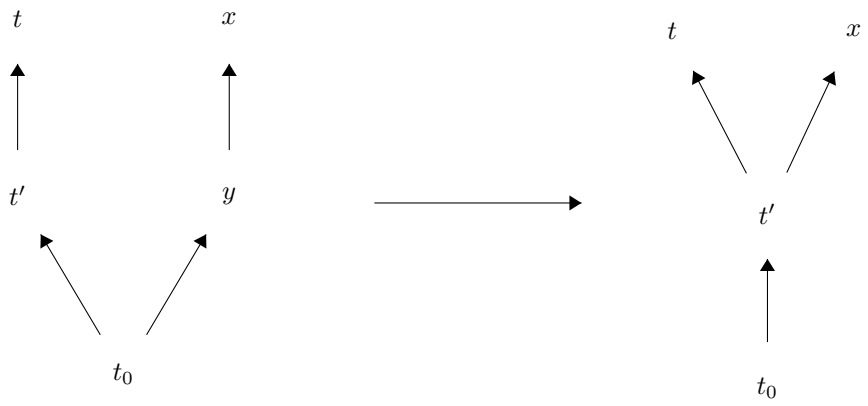
To compute these models, we can use algorithms called the chase. We will present in this document the oblivious chase, the restricted chase and then the core chase. The last chase is the best in the sense of it terminates if and only if there exists a finite universal model.

Nevertheless, it is the slowest so it it is never used for practical applications.

Therefore, we will focus on a restricted type of knowledge base $O = (R, F)$ where the set of conjunctive formulas F is ground (that is there is no variable in the formula) and where the rule contained in R are Horn-$\mathcal{ALC}$ axioms. It is

an interesting restriction because we can represent the results of the chase by a tree.

We will create a quicker chase, that we called the merge chase, that would produce the same output than the core chase. It is based on the idea that in a tree, there exists a brother relation between the nodes (that is two nodes are brothers if they have the same father) and we will see that, in some conditions, a brother of a node can be merged on this node like in the figure below where y is a brother of t' and is merged on it.



We will then try to extend the merge chase on other type of knowledge bases.

Contents

2 Background

We only define what we need in first order logic but we do not redefine basics (interpretations, formulas,...).

2.1 Facts

2.1.1 Syntax

We considered a set of variables **Vars** (often noted $x, y, x_1, \ldots$), a set of constants **Csts** (often noted $a, b, c, c_1, \ldots$), and a set of predicates **Preds** ($P, Q, R, P_1, \ldots$). **Csts**, **Vars**, and **Preds** are pairwise disjoint. A *term* (often noted $t, t_1, \ldots$) is a variable or a constant. We note **Terms** the set of terms. We write $Ar(P)$ to denote the arity of the predicate P.

Definition 2.1. If $t_1, \ldots, t_n$ are terms and P is a predicate with $Ar(P) = n$, then $P(t_1, \ldots, t_n)$ is an *atom*. The atom $P(t_1, \ldots, t_n)$ is *ground* if $t_1, \ldots, t_n$ are constants.

Definition 2.2. A *factbase* F is an existentially closed conjunction of atoms, that is a formula of the form $\exists x_1, \ldots, x_n.P_1(t_1^1, \ldots, t_{k_1}^1) \wedge \ldots \wedge P_m(t_1^m, \ldots, t_{k_m}^m)$ where t_i^j are terms and P_i are predicates. A factbase is *ground* if each of its atoms is ground.

In the introduction, we talked about conjunctive formulas. We can show that a conjunctive formula is always equivalent to a factbase.

In some articles, the factbase are always considered as ground but in this document, we consider factbases that may not be ground. Consequently, a boolean conjonctive query will be a factbase, so we will only talk about factbases and not introduce the notion of query.

For convenience, we identify factbases as sets of atoms, which allows to use set notions such as set inclusion. For example, we identify the factbase $\exists x, x_1, x_2, x_3.P(x) \wedge Q(x, a) \wedge R(x_1, x_2, x_3, b)$ with the set of facts $\{P(x), Q(x, a), R(x_1, x_2, x_3, b)\}$.

For a formula A, let $\boldsymbol{Vars}(A)$ (respectively $\boldsymbol{Csts}(A)$, and $\boldsymbol{Terms}(A)$) be the set of variables (resp. constants, and terms) that occur in A.

2.1.2 Semantics

Definition 2.3. A factbase F *entails* another factbase F' (often noted $F \models F'$) if each interpretation satisfying F satisfies F'.

2.1.3 Homomorphism

Definition 2.4 (Substitution)**.** A *substitution* $\sigma : X \to \mathbf{Terms}$ is a function where X is a set of variables. For example $\{x \mapsto z, y \mapsto a\}$ is a substitution from $\{x, y\}$ to **Terms**. By extension:

- if $c \in \mathbf{Csts}$, then $\sigma(c) = c$;
- if $x \in \mathbf{Vars} \setminus X$, $\sigma(x) = x$;
- if $f = P(t_1, \ldots, t_n)$ is an atom, then $\sigma(f) = P(\sigma(t_1), \ldots, \sigma(t_n))$; and
- if $F = \{f_1, \ldots, f_n\}$ is a factbase, then $\sigma(F) = \{\sigma(f_1), \ldots, \sigma(f_n)\}$.

Definition 2.5. For two factbases F and F', a *homomorphism* from F to F' is a substitution $\sigma : var(F) \to term(F')$ where $\sigma(F) \subseteq F'$. We say that a variable x is *map* to $\sigma(x)$.

Definition 2.6. For two factbases F and F', an *isomorphism* h from F to F' is a bijective homomorphism where its inverse is an homomorphism from F' to F.

We identify sets of facts that are unique up to isomorphism.
The following theorem has been proved in ([1],theorem 6.2.3).

Theorem 2.1 (Homomorphism Theorem)**.** A factbase F *entails* another factbase Q if and only if there exists a homomorphism from Q to F.

Example 2.1. The factbase $F = \{P(b, a), A(x)\}$ entails the factbase $Q = \{P(x, a), P(y, z)\}$ due to the homomorphism $\{x \mapsto b, y \mapsto b, z \mapsto a\}$.

Remark 2.1. Given two factbases F and Q, the problem to know if $F \models Q$ is NP-complete ([4], theorem 7).

2.1.4 Core

For a factbase F, let $id_{|F}$ be the substitution mapping each variable in $\mathbf{Vars}(F)$ to itself.

Definition 2.7. A subset $F' \subseteq F$ is a *retract* of F if $F' \models F$. A *retractation* from F to F' is a homomorphism σ from F to F' such that $\sigma_{|F'} = id_{|F'}$.

Proposition 2.1. *F' is a retract of F if and only if $F' \subseteq F$ and there exists a retractation from F to F'.*

Definition 2.8. If a factbase F does not contain a strict retract, then we say that F is a *core*. A *core* of a factbase F (noted *core*(F)) is a minimal retract of F that is a core.

Proposition 2.2. *The cores of a finite factbase F are unique up to isomorphism.*

Hence, we speak of "the" core of a factbase.

Example 2.2. $F_1 = \{B(x, y), R(y, z)\}$ is the core of $F = \{B(x, y), R(y, z), B(x, w), R(w, z)\}$ because:

- $F_1 \subseteq F$;
- $\{x \mapsto x, y \mapsto y, z \mapsto z, w \mapsto y\}$ is a homomorphism from F to F_1, so F_1 is a retract of F;
- all strict subsets of F_1 are not retracts of F_1.

$F_2 = \{B(x, w), R(w, z)\}$ is also the core of F and is indeed isomorphe to F_1 due to the homomorphism $\{x \mapsto x, y \mapsto w, z \mapsto z\}$.

2.2 Existential rules

2.2.1 Syntax

Definition 2.9. Let $\vec{x}$, $\vec{y}$, and $\vec{z}$ be some tuples of variables that are pairwise disjoint. An *(existential) rule* α is a first-order formula of the form

$$\forall \vec{x}.\forall \vec{y}.(A(\vec{x}, \vec{y}) \rightarrow \exists \vec{z}.B(\vec{x}, \vec{z}))$$

where A and B are conjunctions of atoms. We define $body(\alpha) = A$ and $head(\alpha) = B$. We also note $ev(\alpha)$ the set $\vec{z}$ of existential variables of the rule.

We omit the universal quantifiers when representing existential rules.

Definition 2.10. A *knowledge base* O is a pair (R, F) where R is a set of existential rules and F is a ground factbase.

2.2.2 Semantics

Definition 2.11 (Entailment)**.** A factbase F *entails* a rule α if each interpretation satisfying F satisfies α. We will note $F \models R$ if F entails each rule of the rule set R.

Theorem 2.2. A factbase F *entails* a rule $\alpha = A(\vec{x}, \vec{y}) \rightarrow \exists \vec{z}.B(\vec{x}, \vec{z})$ if and only if for every homomorphism σ from A to F, there exists an extension of σ that is a homomorphism from B to F.

Definition 2.12 (Universal model)**.** A factbase M is a *model* for a knowledge base $O = (R, F)$ if $M \models F$ and $M \models R$. A model U for a knowledge base O is *universal* if for every model M of O, $M \models U$.

Example 2.3. We pose $O = (\{\alpha\}, F)$ where $\alpha = A(x) \rightarrow \exists z.R(x, z) \wedge A(z)$ and $F = \{A(b)\}$. $U = \{A(b), R(b, x_0)\} \cup \{A(x_i) \mid i \in \mathbb{N}\} \cup \{R(x_i, x_{i+1}) \mid i \in \mathbb{N}\}$ is a universal model of O. This knowledge base does not admit finite universal models.

Definition 2.13 (Entailment)**.** A knowledge base O *entails* a factbase B (often noted $O \models B$) if for each model M of O, $M \models B$.

Proposition 2.3. *A knowledge base O entails a factbase B if there exists a universal model U such that $U \models B$.*

An important problem that this document has to deal with is: Given a knowledge base $O = (R, F)$ and a factbase Q, does $O \models Q$? It is well-known that this problem is undecidable ([2], theorem 4).

2.3 The chase

The process of applying rules on a factbase in order to infer more knowledge is called forward chaining. Forward chaining in existential rules is usually achieved via a family of algorithms called the chase. It can be seen as a two-steps process. It first repeatedly applies rules to the set of facts (and eventually computes sometimes the core to supress redundant facts). Then it looks for an answer to the query in this saturated set of facts. This saturated set of facts is a universal model of the knowledge base. The chase is sound and complete; so it must be non-terminating since the problem of entailment is undecidable.

To determine how we apply a rule to a set of fact, we introduce the notion of trigger:

Definition 2.14 (Trigger)**.** Let T be a rule set, α be a rule, σ be a substitution, and F be a factbase. The tuple $t = (\alpha, \sigma)$ is an *oblivious trigger* for F (or α is *applicable* on F via σ) if:

- the domain of σ is the set of all variables occurring in $Body(\alpha)$.
- $\sigma(Body(\alpha)) \subseteq F$.

The tuple $t = (\alpha, \sigma)$ is a *restricted trigger* if t is an oblivious trigger and if for all $\hat{\sigma}$ that extend σ over $\mathbf{Vars}(Head(\alpha))$, $\hat{\sigma}(Head(\alpha)) \nsubseteq F$.

Notice that a restricted trigger is also an oblivious trigger. So future definitions that are dealing with oblivious trigger, deal also with restricted trigger.

The chase will considere oblivious triggers to infer new knowledge from an initial factbase. We explain now how it would apply an oblivious trigger, giving rise to the notion of application.

Definition 2.15 (application)**.** Let $t = (\alpha, \sigma)$ be an oblivious trigger of the factbase F. We pose σ^s the substitution that extends σ over $\mathbf{Vars}(Head(\alpha))$ such that for $y \in ev(R)$, $\sigma^s(y) = y_t$ where y_t is a fresh variable unique with respect to the oblivious trigger t and the variable y. The factbase $\mathbf{appl}(F, t) = F \cup \sigma^s(Head(\alpha))$ is called an *application* on the factbase F through the oblivious trigger $t = (\alpha, \sigma)$.

Example 2.4. . If $\alpha = A(x, y) \rightarrow \exists z.B(x, z)$, $F = \{A(b, c)\}$, and $\sigma = \{x \mapsto b, y \mapsto c\}$ then (α, σ) is a restricted trigger for F. $\mathbf{appl}(F, (\alpha, \sigma)) = \{A(b, c), B(b, z_{(\alpha,\sigma)})\}$ where $z_{(\alpha,\sigma)}$ is a fresh variable.

Definition 2.16 (Derivation)**.** An *oblivious derivation* (respectively a *restricted derivation*) from a knowledge base $O = (F, R)$ is a (possibly infinite) sequence $D = F_0, t_1, F_1, t_2, F_2, \ldots$ where $(F_i)_{i\in\mathbb{N}}$ are factbases such that $F_i \subsetneq F_{i+1}$, t_i are oblivious triggers (resp. restricted triggers), $F_0 = F$, and for $i > 0$, $F_i = \mathbf{appl}(F_{i-1}, t_i)$ is obtained by an application.

Definition 2.17 (Fairness)**.** The oblivious (resp. restricted) derivation $D = F_0, t_1, F_1, t_2, F_2, \ldots$ is *fair* if for every i and every oblivious (resp. restricted) trigger t applicable on F_i, there exists $k \geq i$ such that $\mathbf{appl}(F_k, t_k) = F_k$ (resp. t is not anymore a restricted trigger on F_k).

A fair derivation garantees that we consider every possible application. An easy way to have a fair derivation is to do a breadth-first search (BFS) on the terms.

We will now define the oblivious and restricted chase, It is defined in [3].

Definition 2.18. An *oblivious chase* (resp. a restricted chase) for a knowledge base $O = (F, R)$ is a fair oblivious (resp. restricted) derivation $D = F_0, t_1, F_1, t_2, F_2, \ldots$ $F_0 \subseteq F_1 \subseteq F_2 \subseteq \ldots$ so we can pose $Obl(O) = \cup_{i\in\mathbb{N}} F_i$ (resp. $Res(O) = \cup_{i\in\mathbb{N}} F_i$). We say that the oblivious (resp. restricted) chase *terminates* if $Obl(O)$ (resp. $Res(O)$) is finite.

We can notice that the result of the oblivious chase on a knowledge base is unique up to isomorphism whereas the result of the restricted chase really depends on the order of the application of the rules.

The oblivious chase can do a lot of applications that are useless:

Example 2.5. Suppose that we have the knowledge base $O = (\{\alpha\}, F)$ where $\alpha = A(x, y) \rightarrow \exists z.A(y, z) \wedge A(z, y)$ and $F = \{A(a, b)\}$. An oblivious chase derivation is $F_0, t_1, F_1, t_2, F_2, ...$ where $F_0 = F$, $t_1 = (\alpha, \{x \mapsto a, y \mapsto b\})$, $F_1 = \{A(a, b), A(b, z_{t_1}), A(z_{t_1}, b)\}$, $t_2 = (\alpha, \{x \mapsto b, y \mapsto z_{t_1}\})$, $F_2 = \{A(a, b), A(b, z_{t_1}), A(z_{t_1}, b), A(z_{t_1}, z_{t_2}), A(z_{t_2}, z_{t_1})\}$, ... It will never terminate because each new atom brings new rule applications. So the oblivious chase does not terminate on O whereas the restricted chase terminates. A restricted chase derivation is F_0, t_1, F_1 where $F_0 = F$, $t_1 = (\alpha, \{x \mapsto a, y \mapsto b\})$, and $F_1 = \{A(a, b), A(b, z_{t_1}), A(z_{t_1}, b)\}$. This derivation is fair because there is not anymore any restricted trigger for F_1. We have $Res(O) = F_1$.

The oblivious chase is called this way because it can make naive applications (that is $F_i \vDash F_{i+1}$): in the previous example, $F_1 \models F_2$. The restricted chase is less naive because a restricted trigger is applied only if it really adds information (that is $F_i \nvDash F_{i+1}$).

It is well known that:

Theorem 2.3. For a knowledge base O, $Obl(O)$ and $Res(O)$ are universal models of O.

2.3.1 The core chase

It has been firstly defined in [5].

Definition 2.19 (Core derivation)**.** A *core derivation* for a knowledge base $O = (R, F)$ is a (possibly infinite) sequence $D = F_0, F_1, F_2, \ldots$ where $F_0 = F$, and for $i > 0$, either $F_i = \mathbf{appl}(F_{i-1}, t_i)$ is obtained by an application with t_i a restricted trigger, or F_i is obtained by computing the core of F_{i-1}.

Definition 2.20 (Fairness)**.** A core derivation $D = F_0, F_1, F_2, \ldots$ is *fair* if:

- For every i, for every restricted trigger t applicable on F_i, there exists $k > i$ such that t is not anymore a restricted trigger on F_k.
- For every i, there exists $k \geq i$ such that F_k is a core.

Definition 2.21. A *core chase* for a knowledge base $O = (R, F)$ is a fair core derivation $D = F_0, F_1, F_2, \ldots$ The core chase *terminates* on O if there exits a derivation $D = F_0, F_1, F_2, \ldots$ and $i \in \mathbb{N}$ such that there is not anymore any restricted trigger applicable on F_i. In this case, we pose $C(O) = F_i$. Otherwise, if the core chase does not terminate, $C(O)$ is undefined.

We can notice that the result of the core chase on a knowledge base is unique up to isomorphism

The following theorem has been proven in ([5], theorem 7)

Theorem 2.4. The knowledge base $O = (R, F)$ admits a finite universal model if and only if the core chase algorithm terminates on O.

There exists knowledge bases where the restricted chase does not terminate whereas the core chase terminates.

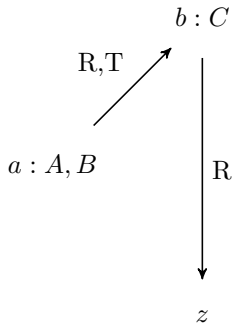
Example 2.6. Suppose that we have the knowledge base $O = (\{\alpha\}, F)$ where $\alpha = A(x, y) \rightarrow \exists z.(A(x, x) \wedge A(y, z))$ and $F = \{A(a, b)\}$. A restricted chase derivation is $F_0, t_1, F_1, t_2, F_2, ...$ where $F_0 = F$, $t_1 = (\alpha, \{x \mapsto a, y \mapsto b\})$, $F_1 = F_0 \cup \{A(a, a), A(b, z_{t_1})\}$, $t_2 = (\alpha, \{x \mapsto b, y \mapsto z_{t_1}\})$, $F_2 = F_1 \cup \{A(b, b), A(z_{t_1}, z_{t_2})\}$, $t_3 = (\alpha, \{x \mapsto z_{t_1}, y \mapsto z_{t_2}\})$, $F_3 = F_2 \cup \{A(z_{t_1}, z_{t_1}), A(z_{t_2}, z_{t_3})\} \ldots$ It will never terminate because each new atom brings new restricted triggers. The core chase terminates on O: a core chase derivation is F_0, F_1, F_2, F_3 where $F_0 = F$, $t_1 = (\alpha, \{x \mapsto a, y \mapsto b\})$, $F_1 = \mathbf{appl}(F_0, t_1) = F_0 \cup \{A(a, a), A(b, z_{t_1})\}$, $t_2 = (\alpha, \{x \mapsto b, y \mapsto z_{t_1}\})$, $F_2 = \mathbf{appl}(F_1, t_2) = F_1 \cup \{A(b, b), A(z_{t_1}, z_{t_2})\}$, $F_3 = \mathit{Core}(F_2) = \{A(a, a), A(a, b), A(b, b)\}$. There is not anymore any restricted trigger for F_3 so the derivation is fair. Consequently the core chase terminates on O.

The core chase always terminates when there exists a finite universal model but this core chase is very expensive in time and it is dificult to define the result of the algorithm when there is no finite universal models because in a core chase derivation $D = F_0, F_1, F_2, \ldots$, we do not necessarily have $F_i \subseteq F_{i+1}$.

3 The merge chase

Definition 3.1. For a factbase F and a term t, we note $\boldsymbol{Preds}_F^1(t)$ the set of unary predicates P such that $P(t) \in F$. For two terms t and t', we note $\boldsymbol{Preds}_F^2(t, t')$ the set of binary predicates P such that $P(t, t') \in F$.

We restricted us to predicates of arity one or two. Hence, we will represent a database F by a labelled graph $G = (V, E)$ where $V = \{t : A_1, \ldots, A_n / t \in \mathbf{Terms}$ and $A_1, \ldots, A_n$ are exactly the elements in $\mathbf{Preds}_F^1(t)\}$ and $E = \{(t_1, t_2)/t_1, t_2 \in \mathbf{Terms}$ and $\mathbf{Preds}_F^2(t_1, t_2) \neq \emptyset$. In this case, we label the edge with exactly the elements in $\mathbf{Preds}_F^2(t_1, t_2)\}$. For example with $F = \{A(a), B(a), R(a, b), T(a, b), C(b), R(b, z)\}$:



3.1 Horn-$\mathcal{ALC}$

Horn-$\mathcal{ALC}$ has been introduced in [6]

Definition 3.2 (Horn-$\mathcal{ALC}$ axioms)**.** A *Horn-$\mathcal{ALC}$ axiom* is an existential rule of the form:

$$A_1(x) \wedge A_2(x) \rightarrow B(x) \tag{1}$$
$$A(x) \wedge R(x, y) \rightarrow B(y) \tag{2}$$
$$A(x) \rightarrow \exists y.R(x, y) \wedge B(y) \tag{3}$$
$$R(x, y) \wedge B(y) \rightarrow A(x) \tag{4}$$

We fix $O = (R, F)$ a knowledge base for this section where R is a Horn-$\mathcal{ALC}$ rule set and F is a ground factbase with predicates of arity one or two.

We can notice that all the variables are introduced by a Horn-$\mathcal{ALC}$ axiom of the form (3).

Definition 3.3. Let F' be a factbase that occured in a chase derivation of the knowledge base O. For a term t_1 and a variable x_2 appearing in F', we say that $t_1 \prec x_2$ if there exists a restricted trigger $tr = (A(x) \rightarrow \exists y.R(x, y) \wedge B(y), \{x \mapsto t_1\})$ such that $x_2 = y_t$. We write $\prec^+$ to denote the transitive closure of $\prec$.

Concretely, $t_1 \prec x_2$ if x_2 has been introduced by t_1 due to a rule of the form (3). Therefore:

Proposition 3.1. *Let F' be a factbase that occured in a chase derivation of the knowledge base O. For every variable x, there exists exactly one predecessor for $\prec$.*

Proposition 3.2. *In the Horn-$\mathcal{ALC}$ theory, let $D = F_0, F_1, \ldots$ be a chase derivation of the knowledge base O. We write $\prec_i$ to denote the relation $\prec$ over the factbase F_i. $\prec_i^+$ is a strict partial order over the set of variables of F_i.*

Proof. We show it by induction on i.

- F_0 is ground so $\mathbf{Vars}(F_0) = \emptyset$ so the initialisation is true.
- Suppose that $\prec_i^+$ is a strict partial order over the set of variables of F_i.
  - $\prec_{i+1}^+$ is transitive over $\mathbf{Vars}(F_{i+1})$ by construction.
  - * If F_{i+1} is the core of F_i, we just take off some facts so $\prec_{i+1}^+ \subseteq \prec_i^+$ so $\prec_{i+1}^+$ is irreflexive.
    * Otherwise, $F_{i+1} = \mathbf{appl}(F_i, t_i)$ with $t_i = (\alpha, \sigma)$ a restricted trigger.
      · If α is not of the form of the rule 3, then $\prec_{i+1}^+ = \prec_i^+$ so $\prec_{i+1}^+$ is also irreflexive by induction hypothesis.
      · Otherwise, $\alpha = A(x) \rightarrow \exists y.R(x,y) \wedge B(y)$ so $F_{i+1} = F_i \cup \{R(\sigma(x), y_{t_i}), B(y_{t_i})\}$ and $\prec_{i+1} = \prec_i \cup (\sigma(x), y_{t_i})$. So $\prec_{i+1}^+ = \prec_i^+ \cup \{(z, y_{t_i}) \mid z = \sigma(x) \text{ or } z \prec_i^+ \sigma(x)\}$. We can see that y_{t_i} has no successor for the relation $\prec_{i+1}$ so, as $\prec_i^+$ is irreflexive by induction hypothesis, $\prec_{i+1}^+$ is irreflexive.
  - Consequently, $\prec_{i+1}^+$ is a strict partial order over the set of variables of F_{i+1}. So the heredity is true.

□

We have shown in the proof that the oriented graphs $(\mathbf{Vars}(F_i), \prec_i)$ does not contain any cycle. Therefore, with the last proposition and proposition 3.1:

Proposition 3.3. *Let F' be a factbase that occured in a chase derivation of the knowledge base O. $(\mathbf{Vars}(F'), \prec)$ is a forest of trees where x_1 is the father of x_2 if $x_1 \prec x_2$.*

We can now define the notion of tree of a term t that is all the facts containing either t or the $\prec$-sucessor of t:

Definition 3.4 (Tree)**.** Let F' be a factbase that occured in a chase derivation of the knowledge base O. For a term t, we note $Tree_{F'}(t) = \{A(t') \mid A(t') \in F' \wedge t' \in \mathbf{Terms} \wedge (t = t' \vee t \prec t')\} \cup \{R(t', x) \mid R(t', x) \in F' \wedge x \in \mathbf{Vars} \wedge t' \in \mathbf{Terms} \wedge (t = t' \vee t \prec t')\}$

We now define the notion of siblings in order to consider an algorithm.

Definition 3.5 (Siblings)**.** Let F' be a factbase that occured in a chase derivation of the knowledge base O. For two terms t_1 and t_2 such that $t_1 \neq t_2$, t_1 is a *strong sibling* of t_2 if:

- t_2 is a variable,
- $\mathbf{Preds}^1_{F'}(t_2) \subseteq \mathbf{Preds}^1_{F'}(t_1)$,
- there exists a term t such that
  - $\mathbf{Preds}^2_{F'}(t, t_2) \neq \emptyset$, and
  - $\mathbf{Preds}^2_{F'}(t, t_2) \subseteq \mathbf{Preds}^2_{F'}(t, t_1)$.

In this case, we say that t_2 is a *weak sibling* of t_1. Two terms t_1 and t_2 such that $t_1 \neq t_2$ are *siblings* if t_1 is a strong or a weak sibling of t_2.

We imposed that a term can only be a strong sibling of a variable because, we will latter map a weak sibling to a strong sibling and it is not possible to map a constant to another term.

Notice that the definition of siblings can be shorted because, in the Horn-$\mathcal{ALC}$ theory, for a term t, there exists a unique term t' and a unique binary relation R such that $R(t', t) \in F'$. So we can replace the last condition in the definition about the existence of t by: there exists a term t and a predicate R such that $R(t, t_1) \in F'$ and $R(t, t_2) \in F'$. We write this more complex definition because it works in more general cases.

When t is a strong sibling of x in F', we want to say that all the triggers applied on x and on the $\prec$-successor of x can be or have already been applied on t and on the $\prec$-successor of t. Therefore, all the facts containing x and the $\prec$-successor of x are or will be redundant. But we do not want to supress all these facts because we know that if they are not yet in the tree of t, they would be computed. That is this intuition that leads us to define merging:

Definition 3.6 (atomic merging)**.** For a factbase F', a term $t \in \mathbf{Terms}(F')$, a variable $x \in \mathbf{Vars}(F')$, and a substitution h where $h(x) = t$ and for all $y \neq x, h(y) = y$, if t is a strong sibling of x in F', then we say that $h(F')$ is the *atomic merging* of x over t in F'.

The following algorithm is the operation that will replace the computation of the core in the core chase. We will show that for a core chase derivation $D = F_0, ..., F_k$ of O, if we apply our algorithm on F_k, then it computes the core of a factbase that could have been produced by continuing the derivation D. In this sense, it always computes the core or something better than the core.

Definition 3.7 (Merging)**.** Let F' be a factbase that occured in a chase derivation of the knowledge base O.

Algorithm 1: Merge(F'):

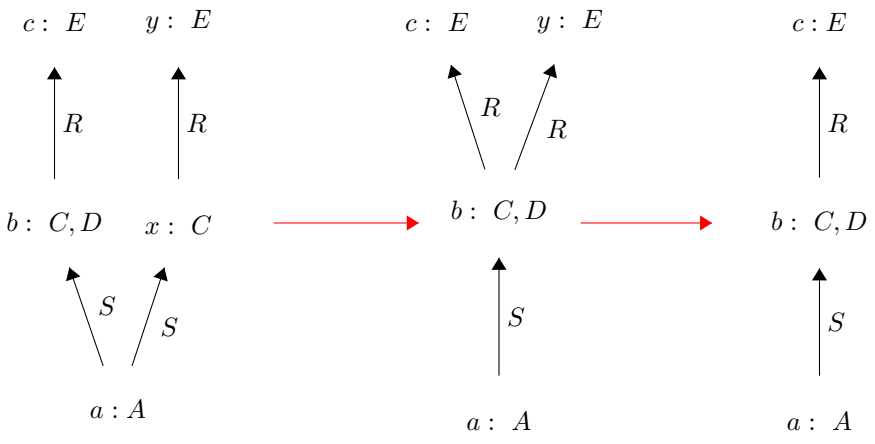
```
1  Let Vars(F') = {x_1, ..., x_n} be such that (x_i ≺^+ x_j) ⇒ i < j ;
2  for i ∈ {1, ..., n} do
3  |  if x_i is still a variable in F' then
4  |  |  for all strong siblings t of x_i do
5  |  |  |  F' ← the atomic merging of x_i over t in F'.
6  |  |  end
7  |  |  for all weak siblings x of x_i such that x is a variable do
8  |  |  |  F' ← the atomic merging of x over x_i in F'.
9  |  |  end
10 |  end
11 end
12 return F'
```

At line 1, we can sort terms like that because, by proposition 3.2, $\prec^+$ is a strict partial order over the set of variables of F'.

We will now consider a new chase:

Definition 3.8. The *merge chase* is the core chase using the merging instead of computing the core, and can be applied only in the Horn-$\mathcal{ALC}$ theory.

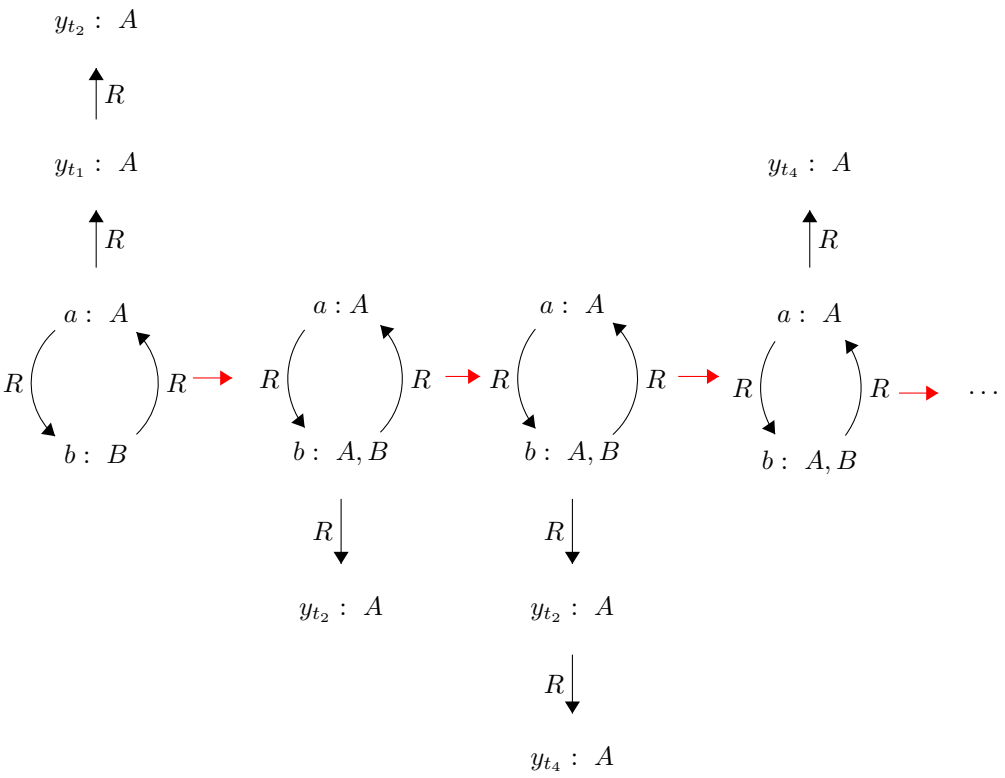
Example 3.1. The figure below is an example of merging. We merge the factbase of the left, $x \prec y$ so we first look the siblings of x and then siblings of y. The variable x have a strong siblings b so we do an atomic merging of x over b that gives us the factbase of the middle. Now, y has a strong sibling c so we do an atomic merging of y over c that gives us the factbase of the right.



We can notice that the order of variables we choose is really important because an atomic merging can create others siblings. In this example, if we treat y before x, then at the the moment where we treat y, it does not have any siblings yet so at the end, we get the factbase of the middle and we will not have merged every possible siblings.

It is really important that the merging does all the possible atomic merging that it can do. Otherwise, the merge chase may not terminates on some knowledge base whereas there exists a finite universal model (like we will see in the next example) and we want that the merge chase terminates on all knowledge base that admits a finite universal model.

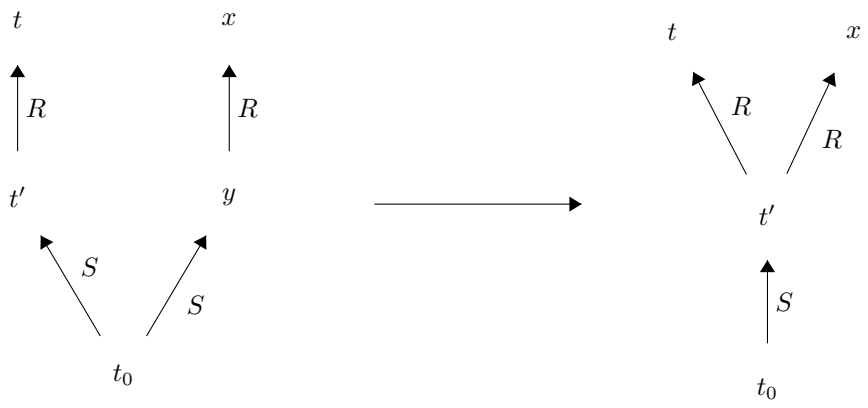
Example 3.2. We consider in the example that $F = \{R(a,b), R(b,a), A(a), A(b)\}$ and $R = \{\alpha, \beta\}$ where $\alpha = A(x) \to \exists y.R(x,y) \wedge A(y)$ and $\beta = B(x) \to A(x)$. We consider the merge chase derivation $F_0 = F, F_1, F_2, ...$ where instead of using the merge operation, we use the atomic merge operation. We applied to F_0 the restricted trigger $t_1 = (\alpha, \{x \mapsto a\})$, we then applied to F_1, the restricted trigger $t_2 = (\alpha, \{x \mapsto y_{t_1}\})$ giving rise to the first factbase of the figure below. We then apply the restricted trigger $t_3 = (\beta, \{x \mapsto b\})$ to obtain the factbase F_3. At this moment, b is a strong sibling of y_{t_1} so we do an atomic merging of y_{t_1} over b to get F_4 that is the second factbase of the figure. F_5 is obtained by the application of the restricted trigger $t_4 = (\alpha, \{x \mapsto y_{t_2}\})$ and is the third factbase of the figure. At this moment, a is a strong sibling of y_{t_2} so we do an atomic merging of y_{t_2} over a to get F_6 that is the last factbase of the figure. We can repeat this infinitely. But O admits a finite universal model: $U = F \cup \{B(b)\}$. So the version of the chase where we consider partial merging is not what we want.



The last example show the importance of doing a total merging. We have to prove that our merging algorithm does a total merging:

Proposition 3.4. *For a factbase G obtained by applying some Horn-$\mathcal{ALC}$ axioms on O, after the merging of G, there does not exists a term t and a variable x such that t is a strong sibling of x.*

Proof. Suppose by contradiction that there exists a term t and a variable x such that t is a strong sibling of x in $Merge(G)$: there exists a term t' and a binary predicate R such that $R(t', x), R(t', t) \in Merge(G)$. This case can happen only if x and t became siblings after that x has been traited by the merging algorithm (and after that t has been traited if t is a variable). Thus, during the merging, there has been an atomic merging over t'. Let y be the variable merged on t' such that $y \prec t$ or $y \prec x$, we suppose without loss of generality that $y \prec x$. We note G^1 the factbase just before the atomic merging of y over t' and we note G^2 the factbase just after the atomic merging. There exists a term t_0 and a binary predicate S such that $S(t_0, t'), S(t_0, y) \in G^1$. G^1 is the left figure and G^2 is the right figure (we do not represent all the graphs):



We have $t_0 \prec t' \prec t$ and $t_0 \prec y \prec x$ so x should have been treated by the algorithm after the merging of t' and y so the algorithm will merge t and x: contradiction. □

We want now prove that a merging compute a core:

Proposition 3.5. *For a factbase G obtained by applying some Horn-$\mathcal{ALC}$ axioms on O, $Merge(G)$ is a core.*

Proof. Suppose by contradiction that $Merge(G)$ is not a core. There exists $G' \subsetneq Merge(G)$ such that G' is a retract of $Merge(G)$. By proposition 1.1, there exists then a retractation h from $Merge(G)$ to G', $var(Merge(G)) \setminus var(G') \neq \emptyset$. Let x be a $\prec$-minimal variable of this set. x is a variable, so has been introduced by the chase due to the axiom 3. So there exists a term t such that $t \prec x$. We have $\mathbf{Preds}^2_{Merge(G)}(t, x) \neq \emptyset$. By $\prec$-minimality of x, $t \in \mathbf{Vars}(G')$. So, as h is a retractation: $h(t) = t$, so for $R \in \mathbf{Preds}^2_{Merge(G)}(t, x)$, $h(R(t, x)) = R(t, h(x)) \in Merge(G)$ and so $R \in \mathbf{Preds}^2_{Merge(G)}(t, h(x))$. Thus $\mathbf{Preds}^2_{Merge(G)}(t, x) \subseteq \mathbf{Preds}^2_{Merge(G)}(t, h(x))$ and $t \prec h(x)$. $x \notin G'$ and $h(x) \in G'$ so $h(x) \neq x$. Let $A \in \mathbf{Preds}^1_{Merge(G)}(x)$. $h(A(x)) \in Merge(G)$ so $A(h(x)) \in Merge(G)$ so $\mathbf{Preds}^1_{Merge(G)}(x) \subseteq \mathbf{Preds}^1_{Merge(G)}(h(x))$. Consequently, $h(x)$ and x are siblings in $Merge(G)$. So, by proposition 3.4, the merging should have suppress x or $h(x)$, so $x \notin Merge(G)$ or $h(x) \notin Merge(G)$: contradiction. So $Merge(G)$ is a core. □

$Merge(G)$ is a core but not necessarily the core of G.

Proposition 3.6. *Let $D = F_0, t_1, F_1, \ldots, t_n, F_n$ be an oblivious chase derivation of the knowledge base O. Let t be a strong sibling of x in F_n and G be the atomic merging of x over t. There exists an oblivious chase derivation $D' = F_0, t_1, F_1, \ldots, t_k, F_k$ of O prolonging D such that $F_k = F_n \cup h(Tree_{F_n}(x))$ where h is a substitution such that $h(x) = t$ and for all $y \neq x, h(y) =$ the variable y where we change x to t .*

mieux définir...

Proof. Let $tr_1 = (\alpha_1, \sigma_1), \ldots, tr_n = (\alpha_n, \sigma_n)$ be all the triggers applied to x and to the $\prec$-sucessor of x ordered by application time. We show by induction on $i \in \{0, \ldots, n\}, H(i)$:

□

3.2 Generalisation

Definition 3.9 (Horn-$\mathcal{ALCH}$ and Horn-$\mathcal{ALCHI}$ axioms)**.** A *Horn-$\mathcal{ALCH}$ axiom* is either a Horn-$\mathcal{ALC}$ axiom or an existential rule of the form:

$$R_1(x, y) \wedge R_2(x, y) \wedge \ldots \wedge R_n(x, y) \rightarrow S(x, y) \tag{5}$$

A *Horn-$\mathcal{ALCHI}$ axiom* is either a Horn-$\mathcal{ALCH}$ axiom or an existential rule of the form:

$$R_1(x, y) \wedge R_2(x, y) \wedge \ldots \wedge R_n(x, y) \rightarrow S(y, x) \tag{6}$$

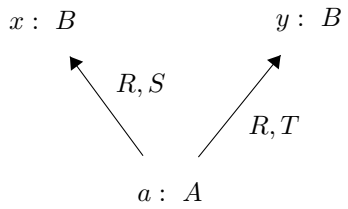
3.2.1 Horn-$\mathcal{ALCH}$

We fix $O = (R, F)$ a knowledge base for this section where R is a Horn-$\mathcal{ALC}$ rule set and F is a ground factbase with predicates of arity one or two. We have to modify the merge chase because it doesn't work anymore:

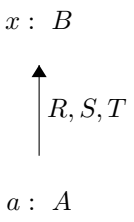
Example 3.3. If we have the knowledge base $O = (R, F)$ where $F = \{A(a)\}$ and the rules are :

$$\begin{aligned} A(x) &\rightarrow \exists y.S(x, y) \wedge B(y) \\ A(x) &\rightarrow \exists y.T(x, y) \wedge B(y) \\ S(x, y) &\rightarrow R(x, y) \\ T(x, y) &\rightarrow R(x, y) \end{aligned}$$

By applying the rules one, two, three and then four, we have the factbase G:



Then, when we apply a merging:



The merging is bad because this factbase is not a universal model of O.

We keep the same relation $\prec$. It is still a strict partial order over the set of variables.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1995.

[2] Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In *Proceedings of the 8th Colloquium on Automata, Languages and Programming*, page 73–85, Berlin, Heidelberg, 1981. Springer-Verlag.

[3] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *Journal of Artificial Intelligence Research*, 48:115–174, Oct 2013.

[4] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, page 77–90, New York, NY, USA, 1977. Association for Computing Machinery.

[5] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In Maurizio Lenzerini and Domenico Lembo, editors, *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 149–158. ACM, 2008.

[6] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexities of horn description logics. *ACM Trans. Comput. Log.*, 14(1):2:1–2:36, 2013.