Implementing the Core Chase for the Description Logic ALC

Maël Abily

June 15, 2021

The goal is to answer a query with a given database and a given set of rules by computing a universal model with an algorithm called the core chase. We are dealing with a restriction of FOL (Horn-$\mathcal{ALC}$ axioms).

Contents

Semantics

1 Background

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

1.1 Facts

1.1.1 Syntax

We considered a set of variables **var** (often noted $x, y, x_1, ...$). We define a *vocabulary* as a tuple (**cst**,**pred**) where **cst** is a set of constants (often noted $a, b, c, c_1, ...$) and **pred**is a set of predicates $(P, Q, R, P_1, ...)$. **cst**, **var** and **pred** are disjoints. A *term* (often noted $t, t_1, ...$) is a variable or a constant. We note **term** the set of terms. We write $Ar(P)$ to denote the arity of the predicate P.

Maybe use a macro? Also, I would make this **Vars** since it is a set. Same for **cst**, **pred**, and *term.*

Discuss: use the Oxford comma.

pairwise disjoint

Definition 1.1. If $t_1, ..., t_n$ are terms and P is a predicate where $Ar(P) = n$, then $P(t_1, ..., t_n)$ is an *atom*. The atom $P(t_1, ..., t_n)$ is said to be *ground* when $t_1, ..., t_n$ are constants.

with

if

Definition 1.2. A *factbase* F is an existentially closed conjunction of atoms: $F := \exists x_1, ..., x_n.P_1(t^1_1, ..., t^1_{k_1}) \wedge ... \wedge P_m(t^m_1, ..., t^m_{k_m})$ where t^j_i are terms and P_i are predicates.

That is, F is a formula of the form . . .

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

Let A be a formula. ***var(A)*** (respectively ***cst(A)***, and ***term(A)***) is the set of variables (resp. constants, and terms) that occur in A. ***FB*** is the set of factbases.

For a formula A, let . . .

1.1.2 Semantic

Semantics

Definition 1.3. A factbase F *entails* a factbase F' (often noted $F \models F'$) if each interpretation satisfying F statisfies F'.

typo

1.1.3 Homomorphism

Definition 1.4 (Substitution)**.** A *substitution* $\sigma : X \rightarrow$ **Term** 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 **Term**. By extension:

- if $c \in$ **cst**, then $\sigma(c) = c$;
- if $x \in$ **var** $\setminus X$, $\sigma(x) = x$;
- if $f = P(t_1, ..., t_n)$ is an atom, then $\sigma(f) = P(\sigma(t_1), ..., \sigma(t_n))$;
- if $F = \{f_1, ..., f_n\}$ is a factbase, then $\sigma(F) = \{\sigma(f_1), ..., \sigma(f_n)\}$

Definition 1.5. Let F and F' be two factbases. A *homomorphism* from F to F' is a substitution $\sigma : var(F) \rightarrow term(F')$ where $\sigma(F) \subseteq F'$.

Definition 1.6. Let F and F' be two factbases. An *isomorphism* $h : F \rightarrow F'$ is a bijective homomorphism where its inverse $h^{-1} : F' \rightarrow F$ is also an homomorphism.

The notation with the arrow is not introduced anymore!

Theorem 1.1 (Homomorphism Theorem)**.** A factbase F *entails* a factbase Q (often noted $F \models Q$) if and only if there exists a homomorphism from Q to F.

Proof. [1] □

Example 1.1. 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 1.1. Given two factbases F and Q, the problem to know if $F \models Q$ is NP-complete.

Add reference here.

Since we can compare two sets of facts with respect to logical consequence, it is natural to consider the core of a set of facts.

Discuss: what is the purpose of this sentence?

1.1.4 Core

Let F be a factbase. $id_{|F}$ is the substitution identity defined by: for all variable $x \in \mathbf{var}(F)$, $id_{|F}(x) = x$.

variables

For a factbase F, let $id_{|F}$ . . .

Definition 1.7. A subset $F' \subseteq F$ is a *retract* of F if there exists a homomorphism $\sigma : F \to F'$ such that $\sigma_{|F'} = id_{|F'}$ (σ is called a *retractation* from F to F').

Definition 1.8. If a factabase F contains a strict retract, then we say that F is *redundant*. Otherwise, it is a *core*. A *core* of a factbase F (noted *core*(F)) is a minimal retract of F that is a core.

Discuss: I am not sure that this word is the most appropriate here; maybe it's best to define what a core is directly.

Proposition 1.1. *The cores of a finite factbase F are unique up to isomorphism. Hence, we speak of "the" core of a factbase.*

Perhaps it makes sense to mention earlier that we identify sets of facts that are unique up to isomorphism. Also, the second sentence does not belong in the proposition.

Example 1.2. $F' = \{B(x,y), R(y,z)\}$ is the core of $F = \{B(x,y), R(y,z), B(x,w), R(w,z)\}$ because:

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

Proposition 1.2. *A factbase F is a core $\Leftrightarrow$ every homomorphism $\sigma : F \to F$ is a bijection.*

Proof. We show it by double-implication.
$\boxed{\Leftarrow}$ By contraposition, suppose that the factbase F is not a core: there exists a strict substet F' of F such that F' is a retract of F. There exists a homomorphism $\sigma : F \to F$ such that $\sigma(F) = F'$. As $F' \subsetneq F$, σ is not surjective, so it is not a bijection.
$\boxed{\Rightarrow}$ Conversely, by contraposition, suppose that there exists an homomorphism σ_1 that is not bijective. As F is finite, σ_1 is not surjective. We pose $F' = \sigma_1(F) \subsetneq F$ and we pose $\sigma_2 : F \to F$ such that for $x \in F'$, $\sigma_2(x) = x$ and for $x \notin F'$, $\sigma_2(x) = \sigma_1(x)$. We have $\sigma_{2|F'} = id_{|F'}$ and $\sigma_2(F) = F'$. So σ_2 is a retractation from F to F' and so F' is a strict retract of F. Consequently F is not a core. □

1.2 Existential rules

1.2.1 Syntax

Definition 1.9. Let $\vec{x}$, $\vec{y}$ and $\vec{z}$ be tuple of variables pairwise disjoint. An *(existential) rule* R is a first-order formula of the form

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

where A and B are conjunctions of atoms. We define $body(R) = A$, $head(R) = B$ and the frontier of R $fr(R) = \vec{x}$ (that is the set of variables shared by the body and the head of R). We also note $ev(R)$ the set $\vec{z}$ of existential variables of the rule.

We will omit the universal quantifiers when representing existential rules.

We omit (tend to use the present.)

Definition 1.10. A *knowledge base* O is a pair (R, F) where R is a set of existential rules and F is a ground factbase.

Definition 1.11. A *Boolean conjunctive query* (or a *query*) is a factbase.

Discuss: maybe we should just talk about factbases?

1.2.2 Semantic

Definition 1.12 (Entailment)**.** A factbase F *entails* a rule α if each interpretation satisfying F statisfies α. We will note $F \models R$ if F entails each rule of the rule set R.

Theorem 1.2. A factbase F *entails* a rule $\alpha = A(\vec{x},\vec{y}) \to \exists \vec{z}.B(\vec{x},\vec{z})$ if and only if for every homomorphism from A to F, there exists an extension of σ that is a homomorphism from B to F.

Note that this homomorphism must only be defined for the variables in A for this to work.

Definition 1.13 (Entailment)**.** Let $O = (R, F)$ be a knowledge base. O *entails* a query B (often noted $O \models B$) if each interpretation satisfying F and satisfying Rstatisfies B.

satisfying F and R also satisfies B

Definition 1.14 (Universal model)**.** A factbase M is a *model* for a knowledge base $O = (T, F)$ 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, there exists a homomorphism $h : U \to M$.

Given a knowledge base $O = (R, F)$, there always exists a model of O that can be considered as a representation of all models of O, ie it is is sufficient to consider this model to check entailment from O. This model has the property of being universal.

Maybe you should introduce the notion of a model earlier so you can use it on this definition.

Example 1.3. We pose $O = (\{\alpha\}, F)$ where $\alpha = A(x) \to \exists z.R(x,z) \wedge A(z)$ and $F = \{A(b)\}$. We pose $U = \{A(b), R(b, x_0)\} \cup \{A(x_i) \mid i \in \mathbb{N}\} \cup \{R(x_i, x_{i+1}) \mid i \in \mathbb{N}\}$. U is a universal model of O. In this knowledge base, all universal models are not finite.

Proposition 1.3. *A knowledge base O entails a query B if $M \models B$ for every model M of O*

An important problem that this document has to deal with is : Given a knowledge base $O = (R, F)$ and a query Q, does $O \models Q$?

It is well-known that this problem is undecidable .

Typo

Discuss: I am not sure that the intuition that you are trying to convey comes through here.

This knowl-

1.3 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 evenually computes sometimes the core to supress redondant 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, and since the problem of entailment is undecidable, this process may not halt.

Definition 1.15 (Trigger)**.** Let T be a rule set, α be a rule, σ be a subsitution and F be a factbase. The tuple $t = (\alpha, \sigma)$ is a *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 an *active trigger* if t is a trigger and if for all $\hat{\sigma}$ that extends σ over $\mathbf{var}(Head(\alpha))$, $\hat{\sigma}(Head(\alpha)) \nsubseteq F$.

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

Definition 1.16 (application)**.** Let $t = (\alpha, \sigma)$ be a trigger of the factbase F. Let $\hat{\sigma}$ be a substitution that extends σ over $\mathbf{var}(Head(\alpha))$ such that for $y \in ev(R)$,$\hat{\sigma}(y) = n$ where n is a newvariable, we note *source*(n)= (α, σ, y). the factbase $\beta(F, t) = F \cup \hat{\sigma}(Head(\alpha))$ is called an *oblivious application* on the factbase F through the trigger $t = (\alpha, \sigma)$. If t is an active trigger, $\beta(F, t) = F \cup \hat{\sigma}(Head(\alpha))$ is called a *restricted application* on the factbase F through the trigger $t = (\alpha, \sigma)$.

Definition 1.17 (Derivation and fairness)**.** A *derivation* from a knowledge base $O = (F, R)$ is a (possibly infinite) sequence of pairs $D = (t_i, F_i)_i$ where $F_0 = F$ and F_{i+1} is obtained by an operation over F_i. We note $(t_j^D)_j$ the sequence of used triggers during the derivation D through the operations application (either oblivious or restricted).The derivation D is *fair* (respectively *actively fair*) if:

- the t_j are pairwise distincts;
- for every i, for every trigger (resp. active trigger) t applicable on F_i, there exists $k > i$ such that t is not anymore an active 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.

Example 1.4. . 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 trigger for F. $\beta(F, (\alpha, \sigma)) = \{A(b, c), B(b, z_0)\}$ where z_0 is a fresh variable such that $source(z_0) = (\alpha, \sigma, z)$.

We will now define the oblivious and restricted chase, It is defined in [2].

Do not write a space before the semi-colon in English

Say “fresh” instead and note with respect to what is unique

I am not sure that it makes sense to talk about restricted and oblivious applications? Aren’t these the same?

Discuss: the last 2 sentences are not very clear...

If you define the naming convention for the variables carefully this point can be simplified by

1.3.1 The oblivious chase

Definition 1.18. An *oblivious chase derivation* for a knowledge base $O = (F, R)$ is a fair derivation $D_{\mathbf{O}} = (t_i, F_i)_i$ where we only use the operation oblivious application. $(F_i)_i$ is monotonic so we can pose $\mathit{Obl(O)} = \cup_{i \in \mathbb{N}} F_i$. We say that the oblivious chase *terminates* if there exists $i \in \mathbb{N}$ such that $F_{i+1} = F_i$. In this case, $\mathit{Obl(O)} = F_i$.

Discuss: this definition should be merged with Def. 1.17.

The oblivious chase is called this way because it forgets to check whether the rule is already satisfied... So we will introduce the restricted chase that is less naive because a trigger is applied only if it is not already satisfied.

1.3.2 The restricted chase

Definition 1.19. A *restricted chase derivation* for a knowledge base $O = (F, R)$ is an actively fair derivation $D_{\mathbf{res}} = (t_i, F_i)_i$ where we use the operation restricted application. $(F_i)_i$ is monotonic so we can pose $\mathit{res(O)} = \cup_{i \in \mathbb{N}} F_i$. We say that the restricted chase *terminates* if there exists $i \in \mathbb{N}$ such that $F_{i+1} = F_i$. In this case, $\mathit{res(O)} = F_i$.

Discuss: this should be merged with Defs. 1.17 and 1.18.

1.3.3 The core chase

It has been firstly defined in [3].

Definition 1.20 (Pruning)**.** Computing the core of the factbase F is an operation called *pruning*. We will explain in the next section, how we calculate it (in the particular case of Horn-$\mathcal{ALC}$ rules).

Definition 1.21 (Core chase)**.** A *core chase derivation* for a knowledge base $O = (T, F)$ is an actively fair derivation $D_{\mathbf{C}} = (t_i, F_i)_i$ where we use the operations restricted application and pruning. The core chase *terminates* on T if there exits $i \in \mathbb{N}$ such that there is not anymore any trigger applicable on T_i. In this case, we pose $C(O) = F_i$. Otherwise, if the core chase does not terminate, $C(O)$ is undefined.

I am not sure that this definition is valid for arbitrary rule sets (that are not Horn-*ALC*.)

Theorem 1.3. The knowledge base $O = (T, F)$ admits a finite universal model if and only if the core chase algorithm terminates on O

Proof. [3] □

1.3.4 Comparaison of the chase algorithms

An oblivious application is fast but the oblivious chase can do a lot of uninteresting applications and does not terminate on some trivial knowledge base.

Example 1.5. If we have the knowledge base $O = (\{\alpha\}, F)$ where $\alpha = A(x, y) \rightarrow \exists z.A(x, z)$ and $F = \{A(a, b)\}$, then $\mathit{Obl(O)} = \{A(a, b)\} \cup \{A(a, x_i) \mid i \in \mathbb{N}\}$ where x_i are new variables. Indeed, the algorithm uses the triggers $(\alpha, \{x \mapsto a, y \mapsto b\})$, $(\alpha, \{x \mapsto a, y \mapsto x_0\})$,$(\alpha, \{x \mapsto a, y \mapsto x_1\})$, etc... So the oblivious algorithm does not stop on O whereas the restricted algorithm stops and $res(O) = F$ because at the initial step, there is only one trigger: $(\alpha, \{x \mapsto a, y \mapsto b\})$ and if we pose $\hat{\sigma} = \{x \mapsto a, y \mapsto b, z \mapsto b\}$ that extends σ, then $\hat{\sigma}(R(x,z)) = R(a,b) \in F$. So this unique trigger is not active.

So the restricted chase is better than the oblivious chase but is less fast. Nevertheless, there exists knowledge bases where the restricted chase does not terminate whereas there exists a finite universal model.

This is actually not the case; I can give you a reference tomorrow.

Example 1.6. If we have the knowledge base $O = (\{\alpha\}, F)$ where $\alpha = A(x,y) \rightarrow \exists z.(A(x,x) \wedge A(y,z))$ and $F = \{A(a,b)\}$, then the restricted chase will use the active triggers : $(\alpha, \{x \mapsto a, y \mapsto b\})$, $(\alpha, \{x \mapsto b, y \mapsto z_0\})$,$(\alpha, \{x \mapsto z_0, y \mapsto z_1\})$, etc... and so $res(O) = \{A(a,a), A(a,b), A(b,b), A(b,z_0)\} \cup \{A(z_i,z_i), A(z_i,z_{i+1}) \mid i \in \mathbb{N}\}$. So the restricted chase does not terminate whereas there exists a universal model $U = \{A(a,a), A(a,b), A(b,b)\}$. The core chase terminates on O: at the first step $F_1 = \{A(a,a), A(a,b), A(b,z_0)\}$. At the second step, if we do an active application, $F_2 = \{(a,a), A(a,b), A(b,b), A(b,z_0), A(z_0,z_1)\}$ (if at this step, we will have done a pruning, $F_2 = F_1$ and we will have continued the chase). If at the third step, we compute the core of F_1, then $F_2 = U$. There is not anymore any active trigger so the core chase terminates.

The core chase always terminates when there exists a finite universal model but this algorithm is very expensive in time and it is dificult to define the result of the algorithm when there is no finite universal models because the computing factbases are not monotonic in comparaison to the factbases computing by the oblivious and restricted chase.

2 Horn-$\mathcal{ALC}$

2.1 Rules

Need to add some references here for Horn-ALC.

Definition 2.1 (Horn-$\mathcal{ALC}$ axioms)**.** A (Horn-$\mathcal{ALC}$) axiom is an existential rule of the form:

$$A_1(x) \wedge ... \wedge A_n(x) \rightarrow B(x) \tag{1}$$

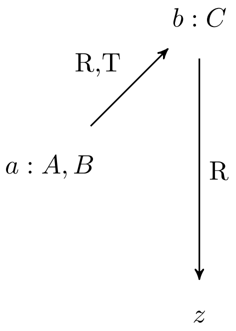
$$A(x) \wedge R(x,y) \rightarrow B(y) \tag{2}$$

$$A(x) \rightarrow \exists y.R(x,y) \wedge B(y) \tag{3}$$

$$R(x,y) \wedge B(y) \rightarrow A(x) \tag{4}$$

Definition 2.2. For a factbase F and a term t, we note $C_F(t)$ the set of unary predicates P such that $P(t) \in F$.

In the Horn-$\mathcal{ALC}$ theory, the rules create only predicates of arity one or two. Hence, we will represent a database F by a labelled graph $G = (V,E)$ where $V = \{t : A_1, ..., A_n / t \in$ **term** and $A_1, ..., A_n$ are exactly the elements in $C_F(t)\}$ and $E = \{(t_1, t_2)/t_1, t_2 \in$ **term** and there exists at least a binary predicate P such that $P(t_1, t_2) \in F$. In this case, we label the edge with exactly the binary predicates P such that $P(t_1, t_2) \in F\}$. For example with $F = \{A(a), B(a), R(a, b), T(a, b), C(b), R(b, z)\}$:



Remark 2.1. We do not represent the predicates of arity greater than 2 in F because they do not really matter.We can also notice that only the edges linking two constants can be labelled with more than one predicate.

I would simply not allow predicates of arity 2 on these fact bases.

Discuss: is this notation really necessary.

2.2 Algorithm

We fix $O = (R, F)$ a knowledge base for this section where R is a Horn-$\mathcal{ALC}$ rule set.

Definition 2.3. Let F' be a factbase that has been genered in a core chase derivation of the knowledge base O. Let t_1, t_2 be two variables appearing in F'. we say that $t_1 \prec t_2$ if there exists a predicate R such that $R(t_1, t_2) \in F'$.We write $\prec^+$ to denote the transitive closure of $\prec$.

Typo.

Shouldn't you consider the case where t_1 is a constant? Also, merge the last 2 sentences into 1.

Proposition 2.1. *In the Horn-$\mathcal{ALC}$ theory, let F' be a factbase that has been genered in a core chase derivation of the knowledge base O. $\prec^+$ is a strict partial order over the set of variables of F'.*

Generated

Discuss: this sentence is somewhat unclear.

Proof.

- $\prec^+$ is transitive by construction
- Suppose by contradiction that there exists a variable x such that $x \prec^+ x$. There exists terms $t_1, ..., t_n$ such that $x \prec t_1 \prec t_2 \prec ... \prec t_n \prec x$. There exists binary predicates $R_0, ..., R_n$ such that $R_0(x, t_1), R_1(t_1, t_2), ..., R_n(t_n, x) \in F'$. We show by induction on $i \in \{0, ..., n\}$, $H(i)$: "for $1 \leq k \leq i, t_k$ is a variable".
  - $H(0)$ is true.
  - Suppose that $H(i-1)$ is true for $i \in \{1, ..., n\}$. We have to show that t_i is a variable. $R_{i-1}(t_{i-1}, t_i) \in F$ and t_{i-1} is a variable, so, as the

initial factbase is ground, $R_{i-1}(t_{i-1}, t_i) \notin F$. So $R_{i-1}(t_{i-1}, t_i)$ has been introduced by the axiom 3. As the application of the core-chase algorithm introduced only variables, t_i is a variable. So $H(i)$ is true.

  - Consequently, $t_1, ..., t_n$ are variables.

Let y be the first variable of the set $\{x, t_1, ..., t_n\}$ introduced by the algorithm. There exists R and $z \in \{x, t_1, ..., t_n\}$ such that $R(z, y) \in F'$. $R(z, y)$ has been introduced by the axiom 3. As the application of the chase algorithm introduced only fresh variables, z is introduced before y. It contradicts the youngness of the variable y. Consequently $x \not\prec^+ x$, so $\prec^+$ is irreflexive over **var**.

□

This proof is correct. However, I reckon that it would be simpler to use an argument via induction from the get go (we can discuss how to do so tomorrow :)

Remark 2.2. We have shown in the proof that the graph $(\mathbf{var}(F'), \prec\ )$ do not contain any cycle. Therefore this graph is a forest of trees.

does

Definition 2.4 (Siblings)**.** Let F' be a factbase that has been genered in a core chase derivation of the knowledge base O. Two terms t_1 and t_2 such that $t_1 \neq t_2$ are *siblings* if $C_F(t_1) \subseteq C_F(t_2)$ or $C_F(t_2) \subseteq C_F(t_1)$ and if there exists a term t and a predicate R such that $R(t, t_1) \in F'$ and $R(t, t_2) \in F'$.

Unclear sentence; discuss.

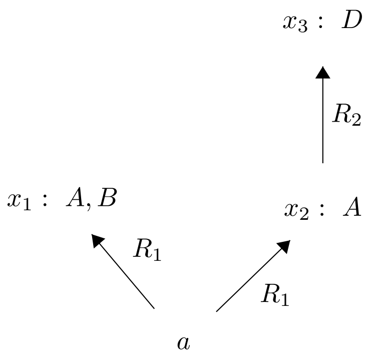
Definition 2.5 ($\mathcal{ALC}$-Pruning)**.** Let F' be a factbase that has been genered in a core chase derivation of the knowledge base O. Let **term(F')** $= \{t_1, ...t_n\}$ is such that $(t_i \in \mathbf{cst} \land t_j \in \mathbf{var}) \Rightarrow i < j$ and $(t_i \prec^+ t_j \land t_i, t_j \in \mathbf{var}) \Rightarrow i < j$ we create an homomorphism h over F'. For all $i \in \{1, ..., n\}$, If t_i is a variable already mapped by h on an other term, we do nothing. Otherwise, we look at all the siblings y of t_i such that y is a variable and $C_{F'}(y) \subseteq C_{F'}(t_i)$. We define $h(z)$ by induction for every z such that $y \prec^+ z$ or $y = z$:

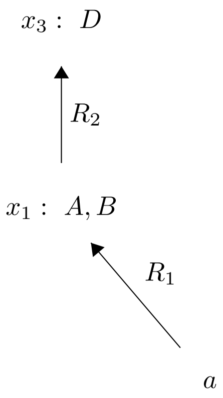
- We pose $h(y) = t_i$.
- If $h(z)$ is defined and there exists z' such that $z \prec z'$, then there exists a binary predicate R such that $R(z, z') \in F'$. There are two cases.
  - If there exists a term t such that $R(h(z), t) \in F'$ and $C_{F'}(t) \subseteq C_{F'}(z')$ or $C_{F'}(z') \subseteq C_{F'}(t)$: We pose $h(z') = t$.
  - Otherwise, we pose $h(z') = z'$.

For all variable x of F' not mapped by h, we pose $h(x) = x$. We pose *prune*(F') $= h(F')$.

We will now consider that the core chase uses the $\mathcal{ALC}$-Pruning instead of the classical pruning.

For example, in the figure below, the $\mathcal{ALC}$-pruning of the factbase of the left gives the factbase of the right due to the homorphism $h = \{x_1 \mapsto x_1, x_2 \mapsto x_1, x_3 \mapsto x_3\}$.





Theorem 2.1. Let F_0 be a factbase that has been genered in a core chase derivation of the knowledge base O. $\mathcal{ALC}$-*Prune*(F_0) is the core of the factbase F_0.

Proof.

- The $\mathcal{ALC}$-pruning algorithm only take off facts of the factbase. Consequently, $prune(F_0) \subseteq F_0$.
- We consider the homorphism $h : F_0 \rightarrow prune(F_0)$ created during the $\mathcal{ALC}$-pruning algorithm. For $x \in \mathbf{var}(F_0)$, suppose that $h(x) \neq x$. According to the $\mathcal{ALC}$-pruning algorithm, x has been erased . So $x \notin \mathbf{var}(prune(F_0))$. So $h_{|Prune(F_0)} = id_{|Prune(F_0)}$. We have shown that h is a retract so $prune(F_0)$ is a retract of F_0.

à préciser

- Suppose by contradiction that $prune(F_0)$ is not a core. There exists $F_0' \subsetneq prune(F_0)$ such that F_0' is a retract of $prune(F_0)$. There exists then a retract $h_1 : Prune(F_0) \rightarrow F_0'$, $var(prune(F_0)) \setminus var(F_0') \neq \emptyset$. Let x be a $\prec^+$-minimal variable of this set. x is a variable, so has been introduced by the $\mathcal{ALC}$-pruning algorithm due to the axiom 3. So there exists a term t and a relation R such that $R(t, x) \in F_0$. By minimality of x, $t \in F_0'$. So $h(t) = t$, so $h(R(t, x)) = R(t, h(x))$. $x \notin F_0'$ and $h(x) \in F_0'$ so $h(x) \neq x$. Let $A \in C_{F_0}(x)$, $x \in var(prune(F_0))$ so $A(x) \in prune(F_0)$. $h(A(x)) \in Prune(F_0)$ so $A(h(x)) \in F_0$ so $A \in C_{F_0}(h(x))$. Consequently, $h(x)$ is a sibling of x in F_0. So the $\mathcal{ALC}$-pruning algorithm should have suppress x or $h(x)$, so $x \notin Prune(F_0)$ or $h(x) \notin Prune(F_0)$: contradiction. So $prune(F_0)$ is a core of F_0

To conclude, *prune(*F_0*)* is a core of the factbase F_0. □

Theorem 2.2. The knowledge base $O = (T, F)$ admits a finite universal model if and only if the core chase algorithm terminates on O

Proof. We have shown that on a factbase of the theory, our $\mathcal{ALC}$-pruning operation was computing a core of the factbase. Consequently, the theorem 2.1 concludes the proof. □

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] 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.

[3] 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.