Implementing the Core Chase for the Description Logic ALC

Maël Abily

June 7, 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-ALC axioms).

Contents

1 Introduction

1.1 Syntax

Definition 1.1 (First-order language, constants, predicates, variables, terms, arity, atoms, facts, factbases)**.** We define a *first-order language* as a set of constants (often noted $a, b, c, c_1, ...$), predicates ($P, Q, R, P_1, ...$), variables ($x, y, x_1, ...$) and nulls which are particular variables. A *term* (often noted $t, t_1, ...$) is a variable or a constant. We note *Ar(P)* the arity of the predicate P. If $t_1, ..., t_n$ are terms and P is a predicate where $Ar(P) = n$, $P(t_1, ..., t_n)$ is an *atom*. A *fact* is a variable-free atom. Let t_i^j be Terms and P_i be predicates. A *factbase* $F := \exists x_1, ..., x_n . P_1(t_1^1, ..., t_{k_1}^1) \wedge ... \wedge P_m(t_1^m, ..., t_{k_m}^m)$ is an existentially quantified conjunctions of atoms which is closed (it means that every variable of F is quantified). *var(F)* (respectively *cst(F)*, *nulls(F)* and *term(F)*) is the set of variables (resp. constants, nulls and terms) that occur in F.*var* (respectively *cst*, *null* and *terms*) is the set of variables (resp. constants, nulls and terms). *Factbases* is the set of factbases.

Discuss: is it really necessary to differentiate nulls and variables?

We write $Ar(P)$ to denote the arity of the predicate P.

Note the use of textit and \$ in the previous comment.

then

Use singular

Remark 1.1. *nulls* $\subseteq$ *var*. *var* and *cst* are disjoints.

Remark 1.2. We identify factbases as sets of atoms. For example, the factbase $\exists x, x_1, x_2, x_3.P(x) \wedge Q(x,a) \wedge R(x_1,x_2,x_3,b)$ is represented by $\{P(x), Q(x,a), R(x_1,x_2,x_3,b)\}$.

Definition 1.2 (Substitution, homomorphism)**.** A *substitution* $\sigma : X \rightarrow Terms$ 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*. We define $\sigma_1 : X \cup const \rightarrow Terms$:

- if $x \in X$, $\sigma_1(x) = \sigma(x)$;
- if c is a constant, $\sigma_1(c) = c$;

We define $\sigma_2 : \{\text{Factbases } F \text{ where } var(F) \subseteq X\} \rightarrow Factbases$:

- if $P(t_1, ..., t_n)$ is an atom in F and $t_1, ..., t_n$ are variables in X or are constants $\sigma_2(\{P(t_1, ..., t_n)\}) = \{P(\sigma_1(t_1), ..., \sigma_1(t_n))\}$;
- if $A_1, ..., A_n$ are atoms in F, $\sigma_2(\{A_1, ..., A_n\}) = \{\sigma_2(\{A_1\}), ..., \sigma_2(\{A_n\})\}$.

Let F and F' be two factbases. A *homomorphism* from F to F' is a substitution $\sigma : var(F) \rightarrow term(F')$ where $\sigma_2(F) \subseteq F'$.

Remark 1.3. We identify σ_2 with σ.

Definition 1.3 (Existential rule, ontology)**.** Let $\vec{x}$, $\vec{y}$ and $\vec{z}$ be tuple of variables. An *(existential) rule* R 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(R)* $= A$, *head(R)* $= B$ and the frontier of R *fr(R)* $= \vec{x}$ (the set of variables shared by the body and the head of R). An *ontology* O is a pair (T, F) where T is a set of existential rules and F is a factbase.

1.2 Core and universal model

Definition 1.4 (Identity, isomorphism, retract, core)**.** $id_{|F}$ is the substitution identity defined by : for all variable $x \in var(F)$, $id_{|F}(x) = x$. An *isomorphism* h is a bijective homomorphism where its inverse is also an homomorphism . A subset $F' \subseteq F$ is a *retract* of F if there exists a subsitution σ such that $\sigma(F) = F'$ and $\sigma_{|F'} = id_{|F'}$ (σ is called a *retractation* from F to F') . If a factabase F contains a strict retract, then we say that F is *redondant*. A factbase is a *core* if it is not redondant. A *core* of a factbase F is a minimal retract of F that is a core.

Remark 1.4. A bijective homomorphism is not necessarily an isomorphism. For example,

$$\sigma : \{R(x)\} \rightarrow \{R(a)\}$$
$$x \mapsto a$$

is a bijective homomorphism but not an isomorphism.

Use something else to represent this sets. E.g., Vars.

"identify with" instead of "identify as"

E.g., we identify the factbase X with the set Y.

Discuss: I believe that these definitions can be simplified to an extent.

Indicate that these tuples are pairwise disjoint

write "that is" right here

F has not been introduced here

I would mention the factbases here explicitly.

Can't you use the notion of a homomorphism to define a retract?

redundant

Otherwise, it is a *core*.

Include discussion: all of the cores of a finite

Example 1.1. $F' = \{B(z), R(x,z)\}$ is the core of $F = \{B(z), B(y), R(x,z), R(x,y)\}$ because :

- $F' \subseteq F$;
- $\{x \mapsto x, y \mapsto z, z \mapsto z\}$ is a retractation from F to F';
- $card(F') = 2$ and $\emptyset, \{B(z)\}, \{B(y)\}, \{R(x,z)\}, \{R(x,y)\}$ are not retracts of F;

Just say "all strict subsets".

Proposal 1.1. *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 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. It concludes the proof. $\square$

that is not bijective

we do not write a

You can remove the last sentence.

or interroga-

Definition 1.5 (Trigger)**.** Let T be a rule set, α be a rule, σ be a subsitution and F be a factbase. The tuple (α, σ) 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(α).
- $\sigma(Body(\alpha)) \subseteq F$.
- For all $\hat{\sigma}$ which extends σ and has its domain equals to the set of all variables occurring in Body(α) and Head(α), $\hat{\sigma}(Head(\alpha)) \nsubseteq F$

Note that "Body" is written with two different formats in these lines. This should not be the case.

The second condition in this conjunction may be safely ignored.

$Tr_\alpha(F)$ is the set of all trigers (α, σ) for F. $Tr_T(F)$ is the set of all trigers (α, σ) for F where $\alpha \in T$ and σ is a substitution.

Example 1.2. If $\alpha = A(x,y) \to \exists z.B(x,z)$, $F = \{A(b,c)\}$ and $\sigma = \{x \mapsto b, y \mapsto c\}$ then (α, σ) is a trigger for F.

Discuss: I would recommend that you use the Oxford comma.

Definition 1.6 (Satisfaction, model, BCQ, universal model)**.** The rule α is *satisfied* by the factbase F if $Tr_\alpha(F) = \emptyset$. The rule set T is *satisfied* by F (noted $F \models T$) if $Tr_T(F) = \emptyset$. A factset M is a *model* for an ontology $O = (T, F)$ if $M \models F$ and T is satisfied by M. A *Boolean conjunctive query (BCQ)* (or a *query*) is a closed formula of the form $\exists x_1, ..., x_n.F(x_1, ..., x_n)$ where F is a conjunction of atoms. A fact set F *entails* a BCQ $B = \exists x_1, ..., x_n.F(x_1, ..., x_n)$ (noted $F \vDash B$) if there exists a substitution σ such that $\sigma(B) \subseteq F$. An ontology O *entails* a BCQ $B = \exists x_1, ..., x_n.F(x_1, ..., x_n)$ (noted $O \vDash B$) if $M \vDash B$ for every model M of O. A model U for an ontology O is *universal* if for every model M of O, there exists a homomorphism $h : U \to M$.

Do you mean "factbase"?

You can just say that a BCQ is a factbase.

Example 1.3. We pose $O = (\{\alpha\}, F)$ where $\alpha = A(x,y) \rightarrow \exists z.A(x,z)$ and $F = \{A(b,c)\}$. We pose $U = \{A(b,c)\} \cup \{A(b,x_i) \mid i \in \mathbb{N}\}$.

- $F \subseteq U$
- $\{\alpha\}$ is satisfied by U
- let M be a model of O. We construct by induction a sequence $(y_i)_{i\in\mathbb{N}}$ of variables such that $A(b,y_n) \in M$:
  - As $F \subseteq M$, $A(b,c) \in M$ and as $\{\alpha\}$ is satisfied by M, there exists a variable y_0 such that $A(b,y_0) \in M$
  - if y_n is defined and $A(b,y_n) \in M$, as $\{\alpha\}$ is satisfied by M, there exists a variable y_{n+1} such that $A(b,y_{n+1}) \in M$

  We then pose

  $$\begin{aligned} h : U &\rightarrow M \\ x_i &\mapsto y_i \end{aligned}$$

  h is a homomorphism from U to M.

I am a bit confused by this example; let's discuss tomorrow.

Consequently, U is a universal model of O.

Proposal 1.2. *A factbase F entails a factbase F' (often noted $F \models F'$) if and only if there exists a homomorphism from F' to F.For example, $F = \{P(b,a), Q(x)\}$ entails $F' = \{P(x,a)\}$ due to the homomorphism $\{x \mapsto b\}$*

You should write "Proposition" instead of "Proposal".

Proposal 1.3. *If there is at least one binary predicate. Given a factbase F and a query Q, the problem of knowing if $F \models Q$ is NP-complete.*

This is a bit more than we needed for this project. Next time, you can ask us if a result is necessary before you set up to write it. (On the other hand, it's good that you understand these results.)

Proof. The size of the problem is $card(term(F)) + card(term(Q))$.

- We choose, as certificate, a homomorphism σ from Q to F. Firstly, the size of the certificate is $card(var(Q)) + card(terms(F))$ which is polynomial in the size of the problem. Secondly, we can check that the certificate σ is a homomorphism in a time which is polynomial in the size of the problem. Therefore, the problem is in NP.
- We make a reduction from 3-COLOR which is known to be NP-complete. Let $G = (V,E)$ be a graph. Let P be a binary predicate. We pose $Q_G = \{P(x,y)/(x,y) \in E\}$ and $K_3 = \{P(c_1,c_2), P(c_1,c_3), P(c_2,c_1), P(c_3,c_1), P(c_2,c_3), P(c_3,c_2)\}$. We have to show that $K_3 \models Q_G \Leftrightarrow G$ is 3-colorable.
  $\boxed{\Rightarrow}$ Suppose that $K_3 \models Q_G$. There exists a substitution $\sigma : Q_G \rightarrow K_3$. We pose :

  $$\begin{aligned} c : V &\rightarrow \{c_1, c_2, c_3\} \\ x &\mapsto \sigma(x) \end{aligned}$$

if $(x,y) \in E$, $P(x,y) \in Q_G$ and so $P(\sigma(x),\sigma(y)) \in K_3$, so $c(x) \neq c(y)$. Therefore, c is a 3-coloration of G.

$\boxed{\Leftarrow}$ Conversely, suppose that G is 3-colorable. Let $c : V \to \{c_1, c_2, c_3\}$ be a coloration of G. c is a substitution from Q_G to K_3. We have to show that $c(Q_G) \subset K_3$. Let $P(x,y)$ be in Q_G. We have $(x,y) \in E$, so $c(x) \neq c(y)$. So $P(c(x),c(y)) \in K_3$. Therefore, c is a homomorphism from Q_G to K_3 and so $K_3 \models Q_G$. It concludes the proof.

$\square$

1.3 Core chase for finite derivation

Definition 1.7 (Application)**.** Let α be an axiom, F be a factbase and T be a ruleset. For a trigger $(\alpha,\sigma) \in Tr_\alpha(F)$, the *application* of (α,σ) to F is $App_{(\alpha,\sigma)}(F) = F \cup \hat{\sigma}(Head(\alpha))$ where $\hat{\sigma}$ extends σ and for all $y \notin var(\sigma)$, $\hat{\sigma}(y) = z_{(\alpha,\sigma)}$ where $z_{(\alpha,\sigma)}$ is a new null.We pose $App_\alpha(F) = \cup_{(\alpha,\sigma)\in Tr_\alpha(F)} App_{(\alpha,\sigma)}(F)$ and $App_T(F) = \cup_{\alpha \in T} App_\alpha(F)$.

Definition 1.8 (Prune)**.** We note *prune(F)* the core of the factbase F. We will explain in the next section, how we calculate it (in the particular case of Horn-ALC rules).

Definition 1.9 (Core chase)**.** A *core chase sequence* for an ontology $O = (T,F)$ is a sequence $(F_n)_{n\in\mathbb{N}}$ of factbases where :

- $F_0 = F$;
- for all $i \in \mathbb{N}^*$, $F_i = Prune(App_T(F_{i-1}))$.

The core chase *terminates* on T if there exits $i \in \mathbb{N}$ such that $F_{i+1} = F_i$. In this case, we pose *Core(O)* $= F_i$.

is a substitution that

Normally, I prefer to be a more precise here and I define exactly how these new variables are "fresh". Let's discuss this tomorrow.

I have noted this before: we need to mention that a finite set of facts admits a unique core (up to isomorphism.)

Briefly remark that that Core(O) is undefined if the core chase does not terminate.

We need to start adding citations; e.g., here you need to reference "The Chase

2 Horn-ALC

2.1 Rules

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

1. $\forall x.A_1(x) \wedge ... \wedge A_n(x) \to B(x)$
2. $\forall x,y.A(x) \wedge R(x,y) \to B(y)$
3. $\forall x.A(x) \to \exists y.R(x,y) \wedge B(y)$
4. $\forall x,y.R(x,y) \wedge B(y) \to A(x)$

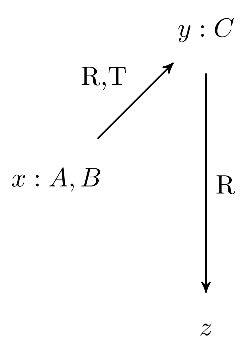
$$\forall x. A_1(x) \wedge ... \wedge A_n(x) \rightarrow B(x) \tag{1}$$
$$\forall x, y. A(x) \wedge R(x, y) \rightarrow B(y) \tag{2}$$
$$\forall x. A(x) \rightarrow \exists y. R(x, y) \wedge B(y) \tag{3}$$
$$\forall x, y. R(x, y) \wedge B(y) \rightarrow A(x) \tag{4}$$

Usually, everyone omits the universal quantifiers in rules.

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

labelled

Remark 2.1. We can then represent a database F by a graph $G = (V, E)$ where $V = \{t : A_1, ..., A_n / t \in \mathit{Terms}$ and $A_1, ..., A_n$ are exactly the elements in $C_F(t)\}$ and $E = \{(t_1, t_2)/t_1, t_2 \in \mathit{Terms}$ 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 $F = \{A(x), B(x), R(x, y), T(x, y), C(y), R(y, z)\}$:



I feel like the definition introduced in the remark is not entirely clear. Let's discuss it tomorrow.

2.2 Algorithm

I consider in this section that all the factbases don't contain variables.

Definition 2.3. Let F be a factbase. Let t_1, t_2 be two nulls appearing in F. we say that $t_1 \blacktriangleleft t_2$ if there exists a predicate R such that $R(t_1, t_2) \in F$. We note $\prec$ the transitive closure of $\blacktriangleleft$.

I would include this restriction on the definition of an ontology. Note that ontologies do not (usually) feature variables in the fact base.

Proposal 2.1. *$\prec$ is a strict partial order over the set of nulls.*

Proof.
- $\prec$ is transitive by construction
- Suppose by contradiction that there exists a null x such that $x \prec x$. There exists terms $t_1, ..., t_n$ such that $x \blacktriangleleft t_1 \blacktriangleleft t_2 \blacktriangleleft ... \blacktriangleleft t_n \blacktriangleleft x$. 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 null".

We write $\prec$ to denote the transitive closure of $\blacktriangleleft$.

- $H(0)$ is true.
- Suppose that $H(i-1)$ is true for $i \in \{1, ..., n\}$. We have to show that t_i is a null. $R_{i-1}(t_{i-1}, t_i) \in F$ and t_{i-1} is a null, so $R_{i-1}(t_{i-1}, t_i)$ has been introduced by the axiom 3. As the application of the chase algorithm introduced only nulls, t_i is a null. So $H(i)$ is true.
- Consequently, $t_1, ..., t_n$ are nulls.

Let y be the first null 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 nulls, z is introduced before y. It contradicts the youngness of the null y. Consequently $x \not\prec x$, so $\prec$ is irreflexive over *null*. □

Remark 2.2. We have shown in the proof that the graph (*null*,◀) don'tcontain any cycle. Therefore this graph is a forest of trees.

Write "do not"

Definition 2.4 (Siblings)**.** 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 $R(t, t_2) \in F$.

This should not be in parenthesis.

I would define the relation sibling only based on ◀. Then discuss when a term is redundant base on another.

Definition 2.5 (Pruning)**.** Let F be a factbase and *terms(F)* $= \{t_1, ...t_n\}$ is such that $(i < j \wedge t_i, t_j \in \mathit{null}) \Rightarrow t_j \not\prec t_i$ The *pruning sequence* of F is the sequence $(F_i)_{i \in \{0,...,n\}}$ of factbases where :

- $F_0 = F$;
- for all $i \in \{1, ..., n\}$,
  - if t_i is not anymore a term of F, $F_i = F_{i-1}$;
  - Otherwise, we look at all the siblings y of x_i such that y is a null. We note S the set containing y and all the nulls z such that $y \prec z$. F_i is the set obtained when we remove every fact of F_{i-1} that contains a null in S.

We pose *prune(F)* $= F_n$.

Lemma 2.1. Let F, F' be two factbases such that $F' \subsetneq F'$ and h a retract from F to F'. $\forall x \in var(F')$, $h(x) = x$.

Proof. Let $x \in var(F')$. There is a predicate P of arity n and terms $t_1, ..., t_n$ such that $P(t_1, ..., t_n) \in F'$ and $x \in \{t_1, ..., t_n\}$. h is a rectract so $h(P(t_1, ..., t_n)) = P(t_1, ..., t_n)$ so for $i \in \{1, ..., n\}$, $h(t_i) = t_i$. In particular, $h(x) = x$. □

Theorem 2.1. Let F be a factbase. *prune(F)* is the core of the factbase F.

Proof.
- The pruning algorithm only take off facts of the factbase. Consequently, $prune(F) \subseteq F$.

- We pose

$$h_0 : F \to prune(F)$$
$$x \mapsto x \text{ if } x \in var(prune(F))$$
$$x \mapsto \text{the unique sibling of } x \text{ which is in } var(prune(F)) \text{ otherwise}$$

  - $h_{0|Prune(F)} = id_{|Prune(F)}$ by construction.
  - For $\alpha \in F$, $h_0(\alpha)$ contains only variable in $prune(F)$, so the pruning algorithm don't take off the fact $h_0(\alpha)$ so $h_0(\alpha) \in prune(F)$. Therefore, $h_0(F) \subseteq prune(F)$. Conversely, as $h_{0|Prune(F)} = id_{|Prune(F)}$, $Prune(F) \subseteq h_0(F)$. So $h_0(F) = prune(F)$

  We have shown that h_0 is a retract so $Prune(F)$ is a retract of F.

- Suppose by contradiction that $prune(F)$ is not a core. There exists $F' \subsetneq prune(F)$ such that F' is a retract of $prune(F)$. There exists then a retract $h_1 : Prune(F) \to F'$. Let $\alpha \in prune(F) \setminus F'$, there are two cases :

  - Case 1 : there exists an unary predicate P and a term t such that $\alpha = P(t)$. If t is a constant, $h_1(\alpha) = \alpha$ and so $\alpha \in Prune(F)$, contradiction : t is not a constant. If $t \in var(F')$, by lemma 2.1, h(t) = t so $h_1(\alpha) = \alpha$ so $\alpha \in F'$: contradiction with $\alpha \notin F'$. Therefore $t \in var(prune(F)) \setminus var(F')$
  - Case 2 : there exists a binary predicate P and two terms t, t_1 such that $\alpha = P(t, t_1)$. If t and t_1 are both constants or in $var(F')$, we have, as in case 1, $h(t) = t$ and $h(t_0) = t_0$, so $h_1(\alpha) = \alpha$ and so $\alpha \in Prune(F)$, contradiction with $\alpha \notin F'$: t or t_1 is in $var(prune(F)) \setminus var(F')$.

  In both cases, we have shown that $var(prune(F)) \setminus var(F') \neq \emptyset$ Let x be a $\prec$-minimal null of this set. x is a null, so has been introduced by the pruning algorithm due to the axiom 3. So there exists a term t and a relation R such that $R(t, x) \in F$. By minimality of x, $t \in F'$. So, by lemma 2.1, $h(t) = t$, so $h(R(t, x)) = R(t, h(x))$. $x \notin F'$ and $h(x) \in F'$ so $h(x) \neq x$. Let $A \in C_F(x)$, $x \in var(prune(F))$ so $A(x) \in prune(F)$. $h(A(x)) \in Prune(F)$ so $A(h(x)) \in F$ so $A \in C_F(h(x))$. Consequently, $h(x)$ is a sibling of x in F. So the pruning algorithm should have suppress all the facts containing x or suppress all the facts containing $h(x)$, so $x \notin Prune(F)$ or $h(x) \notin Prune(F)$: contradiction. So $prune(F)$ is a core of F

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

Theorem 2.2. Let $O = (T, F)$ be an ontology. this ontology admits a finite universal model if and only if the core chase terminates on O

Proof. ... □