An $\mathcal{ALC}$ Description Logic Connection Method

Fred Freitas

Informatics Center - Federal Universidade of Pernambuco (CIn - UFPE)
Av. Prof. Luis Freire, s/n, Cidade Universitária, 50740-540, Recife – PE, Brazil

fred@cin.ufpe.br

Abstract. *The connection method earned good reputation in the field of automated theorem proving for around three decades, due to its simplicity, clarity, efficiency and parsimonious use of memory. This seems to be a very appealing feature, in particular in the context of Semantic Web, where it is assumed that the knowledge bases might be of arbitrary size. In this paper, I present a connection method especially tailored to infer over the description logic (DL) $\mathcal{ALC}$. Our $\mathcal{ALC}$ connection method is formalized in sequent style, although matrices should be employed for practical reasons.*

1. Introduction

The problem of reasoning over ontologies written in Description Logic (DL) [Baader et al 2003] has been receiving strong interest from researchers, particularly since the Semantic Web inception. Regarding this issue, the use of memory is certainly one important asset for a good reasoning performance. I am proposing a formalized inference system which seems adequate to address understandability and the use of a small amount of memory. Our inference system is based on the connection method (CM) [Bibel 1987], which is a simple, clear and effective inference method that has been used successfully over first order logic (FOL). Its main features clearly meet with the demands: (i) it keeps only one copy of each logical sentence in memory; and (iii) it does not derive new sentences from the stored ones.

Definition 1 (Disjunctive normal form (DNF), clause, positive matricial form). A *formula in DNF* is a disjunction of conjunctions, being in the form $C_1 \vee \ldots \vee C_n$, where each C_i is a *clause* (or *dual clause).* Clauses are conjunctions of literals like $L_1 \wedge \ldots \wedge L_m$, also denoted as $\{L_1, \ldots, L_m\}$. Formulae can be also expressed in *disjunctive clausal form* as $\{C_1, \ldots, C_n\}$. Formulae stated this way are also in *positive matricial form*, since they can be represented as a matrix. In the matrix, each clause occupies a column.

Definition 2 (Skolemization). Instead of existential quantifiers, universal quantifiers ($\forall$) are replaced by constants or Skolem functions, since I will work with the whole knowledge based negated (see next section). Variables in the resulting DNF are then (implicitly) existentially quantified.

I started with the Description Logic $\mathcal{ALC}$ (Attributive Concept Language with Complements) [Baader et al 2003], since it constitutes the foundations of many other DLs. I now present an $\mathcal{ALC}$ normal form and the $\mathcal{ALC}$ CM calculus.

2. An $\mathcal{ALC}$ Positive Matricial Normal Form

To reach this normal form, the first two actions to be made over the axioms are: (i) splitting equivalence axioms of the form $C \equiv D$ into two axioms $C \sqsubseteq D$ and $D \sqsubseteq C$, and

(ii) converting all the axioms into a *Negated Normal Form* (NNF), in which negations occurs only on literals [Baader et al 2003]. Next, I define the normal form and impurities with regard to it.

Definition 3 ($\mathcal{ALC}$ disjunction, $\mathcal{ALC}$ conjunction). An $\mathcal{ALC}$ disjunction is either a literal, a disjunction $E_0 \sqcup E_1$ or an universal restriction $\forall r.E_0$. An $\mathcal{ALC}$ conjunction is either a literal, a conjunction $E_0 \sqcap E_1$ or an existential restriction $\exists r.E_0$. E_0 and E_1 are arbitrary concept expressions.

Definition 4 ($\mathcal{ALC}$ pure disjunction). The set S_D of $\mathcal{ALC}$ pure disjunctions is the smallest set where: (i) $D_0 \in S_D$ for every literal D_0; (ii) If $D_0, D_1 \in S_D$, then $D_0 \sqcup D_1 \in S_D$; and (iii) if $D_0 \in S_D$ then $\forall r.D_0 \in S_D$. An element $\breve{D} \in S_D$ is an $\mathcal{ALC}$ pure disjunction. An $\mathcal{ALC}$ *non-pure disjunction* is an $\mathcal{ALC}$ disjunction that is not pure.

Definition 5 ($\mathcal{ALC}$ pure conjunction). The set S_C of $\mathcal{ALC}$ pure conjunctions is the smallest set where: (i) $C_0 \in S_C$ for every literal C_0; (ii) if $C_0, C_1 \in S_C$ then $C_0 \sqcap C_1 \in S_C$; and (iii) if $C_0 \in S_C$ then $\exists r.C_0 \in S_C$. An element $\hat{C} \in S_C$ is an $\mathcal{ALC}$ pure conjunction. An $\mathcal{ALC}$ non-pure conjunction is an $\mathcal{ALC}$ conjunction that is not pure.

Definition 6 (Impurity of a non-pure expression). Impurities of non-pure $\mathcal{ALC}$ DL expressions are either conjunctive expressions in a non-pure disjunction or disjunctive expressions in a non-pure conjunction. The set of impurities is called $\mathcal{ALC}$ *impurity set*, and is denoted by S_I.

Example 1 (Impurities on non-pure expressions).

The expression $(\forall r.(D_0 \sqcup ... \sqcup D_n \sqcup (C_0 \sqcap ... \sqcap C_m) \sqcup (A_0 \sqcap ... \sqcap A_p))$, a non-pure disjunction, contains two impurities: $(C_0 \sqcap ... \sqcap C_m)$ and $(A_0 \sqcap ... \sqcap A_p)$.

Definition 7 (Positive normal form). An $\mathcal{ALC}$ axiom is in positive normal form iff it is in one of the following forms: $(i)\hat{C} \sqsubseteq \breve{D}$; $(ii)C \sqsubseteq \exists r.\hat{C}$; and $(iii)\ \forall r.\breve{D} \sqsubseteq C$; where C is a concept name, $\hat{C}$ a pure conjunction and $\breve{D}$ a pure disjunction.

[Freitas et al 2011] contains $\mathcal{ALC}$ transformation algorithms to this normal form.

2.1. Translation Rules for the normalization

With all axioms in normal form, it is easy to map them both to FOL and to the matricial form, by applying the rules given in Table 1. Table 2 brings the mapping treatment of recursive sub-cases of existential and universal restrictions, when they occur inside any of the three normal forms. An improvement of the approach is, as the usual DL notation, we do not need variables, since all relations are binary.

In order to prove $KB \models \alpha$, the whole knowledge base *KB* is negated during this transformation, once we wish to prove $\neg KB \vee \alpha$ valid. Because of that, subsumption axioms of the form $C \sqsubseteq D$, which are logically translated as $C \rightarrow D$, because negated ($\neg(C \rightarrow D)$, indeed), are now translated to $C \wedge \neg D$, instead of $\neg C \vee D$. Moreover, to establish a uniform set of rules to apply over formulae, we deal with $\neg\alpha$ instead of α, so we consider formulae as $\neg A_1 \vee \ldots \vee \neg A_n \vee \neg\neg\alpha$ where $A_i \in \mathcal{T}$ (axioms in the TBox). The translation rules can then be applied over $\neg\alpha$ and all A_i.

Regarding skolemization, one representational advantage of the approach resides in the clearer matrix representation of universally quantified roles' ($\forall r.C$ or in the matrices, the negated $\exists r.C$). This construct, by definition, has the interpretation $(\forall r.C)^I = \{\forall b, (a,b) \in R^I \rightarrow b \in C^I\}$. Hence, for an axiom of the form $A \equiv \forall r.C$, the definition does not oblige concept A to dispose of instances – this is indeed a very

common error from DL users. But maybe it is not their fault: for instance, tableaux proofs over such axioms don't stress this semantics, in the sense that it allows instances of A without any role instances from r associated to it. In the $\mathcal{ALC}$ CM, the matricial representation explicits this situation: either there are no role instances ($\neg r$) or when it has a role instance (a,b), b has to be an instance of concept C.

Table 1. Translation rules to map $\mathcal{ALC}$ into FOL positive NNF and matrices.

Axiom type	FOL Positive NNF mapping	Matrix
$C \sqsubseteq \exists r.\hat{C}$, where $\hat{C} = \bigcap_{i=1}^{n} A_i$, with $A_i \in S_C$ (pure conjunction)	$(C(x) \wedge \neg r(x,f(x))) \vee (C(x) \wedge \neg A_1(f(x))) \vee \ldots \vee (C(x) \wedge \neg A_n(f(x)))$	$\begin{bmatrix} C & C & \cdots & C \\ \neg r & \neg A_1 & \cdots & \neg A_n \end{bmatrix}$
$\forall r.\breve{D} \sqsubseteq C$, where $\breve{D} = \bigcup_{j=1}^{m} A'_j$, with $A'_j \in S_D$ (pure disjunction)	$(\neg r(x,f(x)) \wedge \neg C(x)) \vee (\neg A'_1(f(x)) \wedge \neg C(x)) \vee \ldots \vee (\neg A'_m(f(x)) \wedge \neg C(x))$	$\begin{bmatrix} \neg r & A'_1 & \cdots & A'_m \\ \neg C & \neg C & \cdots & \neg C \end{bmatrix}$
$\hat{C} \sqsubseteq \breve{D}$, where $\hat{C} = \bigcap_{i=1}^{n} A_i$, $\breve{D} = \bigcup_{j=1}^{m} A'_j$, $A_i \in S_C$ (pure conjunction), $A'_j \in S_D$ (pure disjunction)	$A_1(x) \wedge \ldots \wedge A_n(x) \wedge \neg A'_1(x) \wedge \ldots \wedge \neg A'_m(x)$	$\begin{bmatrix} A_1 \\ \vdots \\ A_n \\ \neg A'_1 \\ \vdots \\ \neg A'_n \end{bmatrix}$

Table 2. Recursive sub-cases of existential and universal restrictions.

Axiom type	FOL Positive DNNF mapping	NNF Positive Matrix	Direct Matrix
A_i is an existencial restriction: $\ldots \sqcap \exists r.A \sqcap \ldots$, with $A \in S_C$ (pure conjunction)	$\ldots \wedge r(x,y) \wedge A(y) \wedge \ldots$	$\begin{bmatrix} \vdots \\ r(x,y) \\ A(y) \\ \vdots \end{bmatrix}$	$\begin{bmatrix} \vdots \\ r \\ A \\ \vdots \end{bmatrix}$
A'_j is an universal restriction: $\ldots \sqcup \forall r.A' \sqcup \ldots$, with $A' \in S_C$ (pure disjunction)	$\ldots \wedge r(x,y) \wedge \neg A'(y) \wedge \ldots$	$\begin{bmatrix} \vdots \\ r(x,y) \\ \neg A'(y) \\ \vdots \end{bmatrix}$	$\begin{bmatrix} \vdots \\ r \\ \neg A' \\ \vdots \end{bmatrix}$

3. An $\mathcal{ALC}$ Connection Calculus in Sequent Style

Definition 3 (Path, connection, unifier, substitution). A *path* is a set of literals from a matrix in which every clause (or column) contributes with one literal. A *connection* is a pair of complementary literals from different clauses, like $\{L_1^\sigma, \neg L_2^\sigma\}$, where $\sigma(L_1)$ (or $\sigma(\overline{L_2})$) is the most general unifier (mgu) between predicates L_1 and $\neg L_2$. σ is the set of

substitutions, which are mappings from variables to terms.

Definition 4 (Validity, active path, set of concepts). An $\mathcal{ALC}$ formula represented as a matrix is *valid* when every path contains a connection $\{L_1, \neg L_2\}$, provided that $\sigma(L_1) = \sigma(\overline{L_2})$. This is due to the fact that a connection represents the tautology $L_1^\sigma \vee \neg L_2^\sigma$ in DNF. As a result, the connection method aims at finding a connection in each path, together with a unifier for the whole matrix. During the proof, the current path is called *active path* and denoted by $\mathcal{B}$. The *set of concepts* τ of a variable or instance *x* during a proof is defined by $\tau(x) \stackrel{\text{def}}{=} \{C | C(x) \in \mathcal{B}\}$ [Schmidt & Tishkovsky 2007].

Definition 5 ($\mathcal{ALC}$ connection sequent calculus). Figure 1 brings the rules in sequent style of the $\mathcal{ALC}$ connection calculus, adapted from [Otten 2010].

$$Axiom\ (Ax)\ \frac{}{\{\}, M, Path}$$

$$Start\ Rule\ (St)\ \frac{C_2, M, \{\}}{\varepsilon, M, \varepsilon}$$

$$where\ M\ is\ the\ matrix\ KB \vDash \alpha,\ C_2\ is\ a\ copy\ of\ C_1 \in \alpha$$

$$Reduction\ Rule\ (Red)\ \frac{C^\sigma, M, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}}\ with\ \sigma(L_1) = \sigma(\overline{L_2})$$

$$Extension\ Rule\ (Ext)\ \frac{C_2^\sigma \backslash \{L_2^\sigma\}, M, Path \cup \{L_1\} \quad C^\sigma, M, Path}{C \cup \{L_1\}, M, Path}$$

$$with\ \ C_2\ a\ copy\ of\ C_1 \in M, L_2 \in C_2, \sigma(L_1) = \sigma(\overline{L_2}),$$

$$Copy\ Rule\ (Cop)\ \frac{C \cup \{L_1\}, M \cup \{C_2^\mu\}, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}}$$

$$with\ L_2 \in C_2, \mu \leftarrow \mu + 1, and\ \left(x_\mu^\sigma \notin N_O\ or\ \tau(x_\mu^\sigma) \nsubseteq \tau\left(x_{\mu-1}^\sigma\right)\right), \sigma(L_1) = \sigma(\overline{L_2})\ (blocking\ conditions)$$

Figure 1. The $\mathcal{ALC}$ connection calculus rules in sequent style (adapted from [Otten 2010]).

Blocking didn't occur in the original CM due to FOL semi-decidability, but it consists in a common practice in DL to guarantee termination. Here, to assure termination, we have to check if the set of concepts τ associated to the variable x_μ^σ (i.e., if the new x_μ was unified) of the new literal L_2^μ being created by the Cop rule is not contained in the set of concepts of the original *x* from $L_2(x)$ (in the rule, $\tau(x^\sigma)$) [Schmidt & Tishkovsky 2007]. Examples of the $\mathcal{ALC}$ CM calculus, as well as an algorithm of the system based on [Bibel 1987] can be found at [Freitas et al 2010].

In terms of complexity, the system is PSPACE in case of non-cyclical ontologies and EXPTIME for cyclical. Proofs of its completeness, soundness and termination are presented in [Freitas et al 2010].

Example 1 ($\mathcal{ALC}$ connection calculus).

$$\left.\begin{array}{l} \text{Animal} \sqcap \exists \text{hasPart.Bone} \sqsubseteq \text{Vertebrate} \\ \text{Bird} \sqsubseteq \text{Animal} \sqcap \exists \text{hasPart.Bone} \sqcap \exists \text{hasPart.Feather} \end{array}\right\} \vDash \text{Bird} \sqsubseteq \text{Vertebrate}$$

In FOL positive matricial clausal form, where the variables *y* and *t* were respectively skolemized by the function *f(x)* and the constant *c*, the formula is represented by

{{*Bird(x)* ,¬*Animal(x)*}, {*Bird(x)* ,¬*hasPart(x,f(x))*}, {*Bird(x)* ,¬*Bone(f(x))*}, {*Bird(x)* ,¬*hasPart(x,g(x))*}, {*Bird(x)* ,¬*Feather(g(x))*}, {*Animal(w), hasPart(w,z), Bone(z),* ¬*Vertebrate(w)*}, {¬*Bird(c)*}, {*Vertebrate(c)*)}}.

Figure 2 deploys the query proof. In the figure, literals of the active path are in boxes and arcs denote connections. For building a proof, we first choose a clause from the consequent (*Start rule*), say, the clause {¬*Bird(c)*} and a literal from it (¬*Bird(c)*).

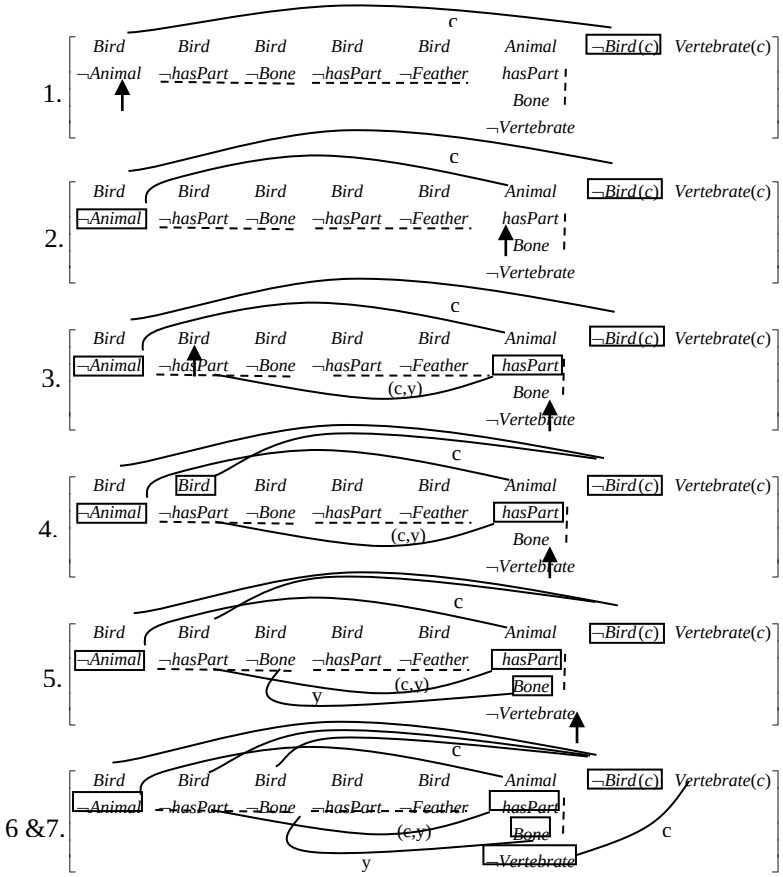

Figure 2. A connection proof example in matricial form.

Step 1 connects this clause with the first matrix clause. An instance or variable - representing a fictitious individual we are predicating about -, appears in each arc, for this connection, the instance *c*. The arrow points to literals to be checked in the clause (¬*Animal* in Step 1), that should be checked afterwards. After step 2, the connection {¬*Animal, Animal*} is not enough to prove all paths stemming from the other clause, the one with literal ¬*Animal*. In order to assure that, the remaining literals from that clause, *viz hasPart, Bone* and ¬*Vertebrate*, have still to be connected. Then, in step 3, when we connect *hasPart,* we are not talking about instance *c* any more, but about a relation between it and another variable or fictitious individual, say *y* (indicated by *(c,y)*).

Until that moment, we were only applying the *Extension rule.* However, in step 4, we use the *Reduction rule*, triggered by its two enabling conditions: (i) there is a connection for the current literal already in the proof; and (ii) unification can take place.

Unification would not be possible if we were referring to different individuals or skolemized functions (in $\mathcal{ALC}$, equality among individuals is not necessary).

A small note on unification is necessary here, because it brings a small trick to the calculus. Since horizontal dashlines represent universal restrictions ($\forall r.C$), the qualifier concept (C, represented as $\neg C$ in the matrix) correspond to a skolemized concept (say *C(f(c))*). Therefore, it can only be unified with variables, but not with concrete individuals or other skolemized qualifier concepts.

In case the system is able to summon the query, the processing finishes when all paths are exhausted and have their connections found. In case a proof cannot be entailed, the system would have tried all available options of connections, unifiers and clause copies, having backtracked to the available options in case of failure.

4. Conclusions and Future Work

I have formalized a connection method to take on the DL $\mathcal{ALC}$, by adapting the CM calculus formalized in sequent style from [Otten 2010] and including a new rule. I also introduced some notational improvements, the key one being the representation without variables. Of course, I plan to continue this work in many research directions, such as implementations, other DLs, Semantic Web, etc.

I intend to extend the work presented here to more complex description logic languages in a near future. Particularly, formalizations and implementations for the DLs $\mathcal{EL}$++, $\mathcal{SHIQ}$ and $\mathcal{SROIQ}$ will be practically useful for applications related to the Semantic Web and for some other biomedical applications that I am involved in.

Last but not least, lean implementations written in Prolog, in the flavor of leanCop [Otten & Bibel 2003], that demand small memory space, can serve applications that are constrained in memory, such as stream reasoning in mobile applications, for instance. They are also in my research agenda.

References

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds.): The Description Logic Handbook. Cambridge University Press, 2003.

Bibel, W. Automated theorem proving. Vieweg Verlag, Wiesbaden, 1987.

Freitas, F. A Connection Method for Reasoning with the Description Logic $\mathcal{ALC}$. Technical report. 2010. www.cin.ufpe.br/~fred/CM-ALCTechRep.doc

Otten, J.Restricting backtracking in connection calculi.AI Comm,23(2-3):159-182 2010.

Otten, J., Bibel, W. *leanCoP: Lean Connection-Based Theorem Proving*. Journal of Symbolic Computation, Volume 36, pages 139-161. Elsevier Science, 2003.

Schmidt, R., Tishkovsky, D. Analysis of Blocking Mechanisms for Description Logics. In Proceedings of the Workshop on Automated Reasoning, 2007.