Cardinality Restrictions Within Description Logic Connection Calculi

Fred Freitas[1(✉)] and Ivan Varzinczak[2]

[1] Informatics Center, Federal University of Pernambuco (CIn - UFPE), Recife, Brazil
fred@cin.ufpe.br
[2] CRIL, University of Artois & CNRS, Arras, France
varzinczak@cril.fr

Abstract. Recently, we have proposed the θ-connection method for the description logic (DL) $\mathcal{ALC}$, the $\mathcal{ALC}$ θ-CM. It replaces the usage of Skolem terms and unification by additional annotation and introduces blocking through a new rule in the connection calculus, to ensure termination in the case of cyclic ontologies. In this work, we enhance this calculus and its representation to take on $\mathcal{ALCHQ}^{=}$, an extended fragment that includes role hierarchies, qualified number restrictions and (in)equalities. The main novelty of the calculus lies in the introduction of equality, as well as in the redefinition of connection to accommodate number restrictions, either explicitly or expressed through equality. The new calculus uses the Eq system, thus introducing substitutivity axioms for each concept or role name. The application of Bibel's equality connections appears here as a first solution to deal with equality.

Keywords: Description logic · Connection method · Inference system
Cardinality restrictions · Role hierarchies · Reasoning

1 Introduction

Particularly after the appearance of the Semantic Web, Description Logic (DL) [1] has attracted growing attention in the Informatics' mainstream, with applications in many areas. The possibility of supplying Web users with query answers obtained by complex, albeit decidable reasoning may constitute the main reason for such interest.

At least in the last two decades, the field of DL reasoning has been taken over by tableaux calculi and reasoners. The DL family of languages has spread to include very expressive fragments such as $\mathcal{SROIQ}$ [15]; cutting-edge reasoning performance was accordingly achieved, with the development of DL-specific optimization techniques.

On the one hand, a clear advantage for tableaux calculi against the growing array of DL constructs - which demand particular treatment during reasoning - may lie in its easy adaptability. Dealing with a new construct may only require conceiving a new tableaux rule, maybe along with some optimization companion.

On the other hand, promising methods may have been neglected in such a scenario, in which the tough competition is often focused on gains through optimizations.

C. Benzmüller et al. (Eds.): RuleML+RR 2018, LNCS 11092, pp. 65–80, 2018.
https://doi.org/10.1007/978-3-319-99906-7_5

Therefore, perhaps there is still room available for "basic research" on DL reasoning, in the sense that other efficient calculi need to be adapted to DL, tuned and tested.

Recently, we have embarked in such an endeavor. Departing from the successful first-order logic (FOL) Connection Method (CM) - whose matrix representation provides a parsimonious usage of memory compared to other methods -, we designed, a first connection calculus for DL, the $\mathcal{ALC}$ θ-CM [6]. It incorporates several features of most DL calculi: blocking (implemented by a new rule in connection calculi), lack of variables, unification and Skolem functions.

Moreover, RACCOON [7], the reasoner which embodied this calculus, displayed surprisingly promising performance for an engine which has no DL optimizations. In most of our benchmarking for $\mathcal{AL}$, $\mathcal{ALE}$ and $\mathcal{ALC}$, it was only clearly surpassed by Konclude [15] (even against FacT ++ [16] and Hermit [8] – see Sect. 5), even considering that these reasoners were designed to face more complex DL fragments than $\mathcal{ALC}$, a disadvantage for them. Nonetheless, this fact corroborates connection calculi as fair, competitive choices for DL ontology querying and reasoning.

In an attempt to extend the expressivity of the ontologies it can cope with, in this work we enhance this calculus and its representation to take on $\mathcal{ALCHQ}^{=}$, an extended fragment that includes role $\mathcal{H}$ierarchies, $\mathcal{Q}$ualified number restrictions and (in)equalities. The main novelty lies in the introduction of (in)equalities, as well as the redefinition of connection to accommodate number restrictions, either explicitly or expressed through equalities. The application of Bibel's eq-connections (equality connections) [4] appears here as a first solution to deal with (in)equalities, although cardinality restrictions do not need equality connections, once, in this case, an equality connects only to an inequality, given a proper θ-substitution for the pair is available. Surely, there are other more efficient solutions to dealing with equality, such as paramodulation [13] and RUE (Resolution and Unifications with Equality) [5], not to speak on the many advanced techniques already applied in the DL setting. The aim of the new $\mathcal{ALCHQ}^{=}\theta$-connection calculus is providing a first solution and roadmap on how to deal with equality and number restrictions, based on its semantics.

The text is organized as follows. Section 2 provides an explanation of the FOL CM. Section 3 introduces $\mathcal{ALCHQ}^{=}$; its normalization is shown in Sect. 4. Section 5 explains our formal connection calculus for $\mathcal{ALCHQ}^{=}$. Section 6 discusses related work on equality handling in FOL and DL. Section 7 concludes the article. The calculus' termination, soundness and completeness are proven in www.cin.ufpe.br/∼fred/RR.pdf.

2 The Connection Method

The connection method has a long tradition in automated deduction. Conceived by W. Bibel in the early 80's, it is a *validity procedure* (opposed to *refutation procedures* like tableaux and resolution), i.e., it tries to prove whether a formula, theorem or query is valid. It consists of a matrix-based deduction procedure designed to be economical in the use of memory, as it is not *generative* as tableaux and resolution, in the sense that it does not create intermediary clauses or sentences during proof search. We explain how it works below, preceded by necessary definitions.

A (first-order) *literal*, denoted by L, is either an atomic formula or its negation. The complement $\neg L$ of a literal L is P if L is of the form $\neg P$, and $\neg L$ otherwise. A formula in *disjunctive normal form (DNF)* is a disjunction of conjunctions (like $C_1 \vee \ldots \vee C_n$), where each *clause* C_i has the form $L_1 \wedge \ldots \wedge L_m$ and each L_i is a literal. The *matrix* of a formula in DNF is its representation as a set $\{C_1, \ldots, C_n\}$, where each C_i has the form $\{L_1, \ldots, L_m\}$ with literals L_i. In the *graphical matrix* representation, clauses are represented as columns.

2.1 Method Representation

Suppose we wish to entail whether $KB \models \alpha$ is valid using a direct method, like the Connection Method (CM). By the Deduction Theorem [3], we must then prove directly $KB \rightarrow \alpha$, or, in other words, if $\neg KB \vee \{\alpha\}$ is valid. This opposes to classical refutation methods, like tableaux and resolution, which builds a proof by testing whether $KB \cup \{\neg\alpha\} \models \bot$. Hence, in the CM, the whole knowledge base KB should be negated, including instantiated predicates, like $A(a)$, where a is a constant or individual. Given $KB = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, α_i being FOL formulae, in this work we define *query* as a matrix $\neg KB \vee \{\alpha\}$ (i.e., $\neg\alpha_1 \vee \neg\alpha_2 \vee \ldots \vee \neg\alpha_n \vee \alpha$) to be proven valid, where α is the query *consequent*. A query represented in this way is said to be in *positive DNF*.

Besides, the effects for a negated KB in a DNF representation are: (i) axioms of the form $E \rightarrow D$ (in DL, $E \sqsubseteq D$) translate into $E \wedge \neg D$; (ii) in a matrix, variables are existentially quantified; (iii) FOL Skolemization works over universally quantified variables, instead of existentially ones; and (iv) the consequent α is not negated.

Example 1 (Query, positive DNF, clause, matrix). The query

$$\{\forall w\, Animal(w) \wedge \exists z(hasPart(w,z) \wedge Bone(z)) \rightarrow Vertebrate(w), \forall x\, Bird(x) \rightarrow Animal(x) \wedge \exists y(hasPart(x,y) \wedge Bone(y)) \wedge \exists v(hasPart(x,v) \wedge Feather(v))\} \models \forall t\, Bird(t) \rightarrow Vertebrate(t)$$

is represented by the following positive DNF matrix and graphical matrix, where variables y, v and t were skolemized by functions $f(x), g(x)$ and constant c (Fig. 1):

$$\{\{Animal(w), hasPart(w,z), Bone(z), \neg Vertebrate(w)\}, \{Bird(x), \neg Animal(x)\}, \{Bird(x), \neg hasPart(x, f(x))\}, \{Bird(x), \neg Bone(f(x))\}, \{Bird(x), \neg hasPart(x, g(x))\}, \{Bird(x), \neg Feather(g(x))\}, \{\neg Bird(c)\}, \{Vertebrate(c)\}\}.$$

$$\begin{bmatrix} A(w) & B(x) & B(x) & B(x) & B(x) & B(x) & \neg B(c) & V(c) \\ h(w,z) & \neg A(x) & \neg h(x,f(x)) & \neg Bo(f(x)) & \neg h(x,g(x)) & \neg F(g(x)) & & \\ Bo(z) & & & & & & & \\ \neg V(w) & & & & & & & \end{bmatrix}$$

Fig. 1. A FOL query in disjunctive clausal form represented as a matrix and graphical matrix (with literals abridged, e.g. $A(w)$ stands for $Animal(w)$, etc.)

2.2 Method Intuition and Functioning

We have represented a FOL query in DNF with clauses as columns, i.e., we are dealing with the matrix vertically. If we change our perspective, traversing the matrix horizontally in all possible ways (or *paths*), with each column supplying only one literal in a path, and group these paths conjunctively, we are indeed converting the query to the conjunctive normal formal (in the most inefficient way). For instance, in the matrix above, two of the paths are (randomly) listed below:

$$\{A(w), B(x), B(x), \neg Bo(f(x)), \neg h(x, g(x)), B(x), \neg B(c), V(c)\}$$
$$\{h(w, z), \neg A(x), B(x), \neg Bo(f(x)), \neg h(x, g(x)), \neg F(g(x)), \neg B(c), V(c)\}.$$

The conjunctive formula would look like (with all variables quantified):

$$\ldots \wedge (A(w) \vee B(x) \vee B(x) \vee \neg Bo(f(x)) \vee \neg h(x, g(x)) \vee B(x) \vee \neg B(c) \vee V(c)) \wedge \ldots$$
$$\wedge (h(w, z) \vee \neg A(x) \vee B(x) \vee \neg Bo(f(x)) \vee \neg h(x, g(x)) \vee \neg F(g(x)) \vee \neg B(c) \vee V(c)) \wedge \ldots$$

It is now easy to see that such a formula (or matrix) is valid iff every path has a connection, i.e., a σ-complimentary pair of literals, where σ is the (most general) unifier between them. For instance, the first path above is true, once it contains the valid subformula $B(x) \vee \neg B(c)$, with $\sigma = \{x/c\}$; the second is true because it has the subformula $h(w, z) \vee \neg h(x, g(x))$, with $\sigma = \{x/c, w/c, z/g(c)\}$, and so on.

The method then must check all paths for connections in a systematic way. Note that a connection prunes many paths in a single pass, due to the matricial arrangement of clauses, a relevant source of reasoning efficiency.

Example 2 (Connection Method). Figure 2 shows the step-by-step query solution. The reader may note, e.g., that the first connection (step 1) solves 16 paths.

Each connection can create up to two sets of literals still to be solved, one in each clause (column) involved in the connection. The first of these literals in each clause is marked in each step of the Figure with an arrow.

Otten [11] proposed a "sequent-style" calculus formalization, alternatively to the graphical matricial one. Our calculus is based on his; it is explained in Sect. 5.

3 The Description Logic $\mathcal{ALCHQ}^{=}$

An ontology in $\mathcal{ALCHQ}^{=}$ is a set of axioms over a signature $\Sigma = (N_C, N_R, N_O)$, where N_C is the *set of concept names* (unary predicate symbols), N_R is the *set of role or property names* (binary predicate symbols), and N_O is the set of *individual names* (constants) [1]. The sets are mutually disjoint. The set of $\mathcal{ALCHQ}^{=}$ *concept expressions* ($\mathbf{C}$) is recursively defined as follows (with $n \in \mathbb{N}^*$, and C a concept expression, i.e., $C \in \mathbf{C}$):

$$C ::= N_C | C \sqcap C | C \sqcup C | \neg C | \exists r.C | \forall r.C | \geq nr | \geq nr.C | \leq nr | \leq nr.C$$

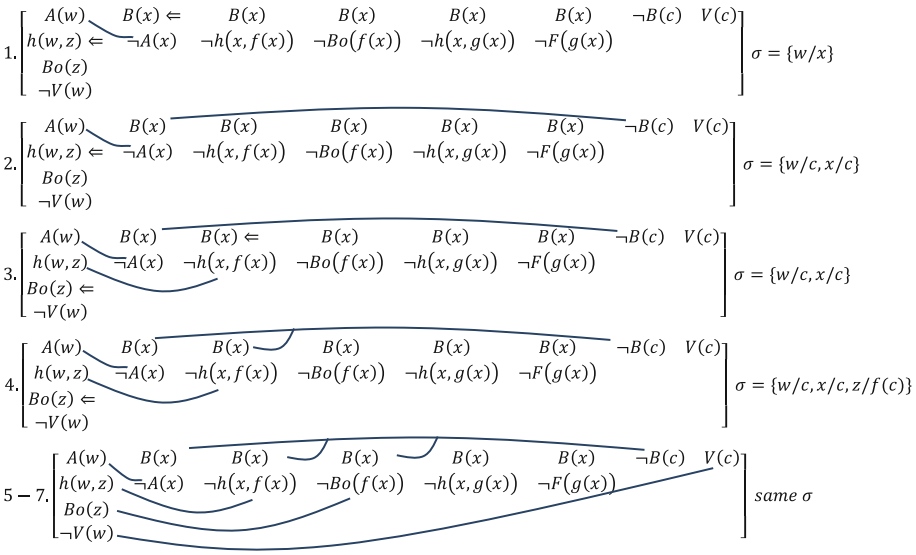

Fig. 2. The query solution, with literals abridged. Arrows stand for pending sets of literals.

$\mathcal{ALCHQ}^{=}$ allows for a set of basic axioms (TBox, RBox), and a set of axioms of a particular situation (ABox). In the definitions below $a, b \in N_O, r, s \in N_R, D, E \in \mathbf{C}$ and $i, n \geq 1$. A TBox axiom is a subsumption like $D \sqsubseteq E$; an RBox one is like $r \subseteq s$; and an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$, an RBox $\mathcal{R}$ is a finite set of assertions (or *instances*) of three types: (i) *concept assertions* like $C(a)$; (ii) *role assertions* $r(a, b)$; (iii) *(in)equality assertions* $a = b$ (or $a \neq b$). An ontology O is an ordered tuple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$

An interpretation I has a domain Δ^I and an interpretation function $.^I$ that maps to every $A \in N_C$ a set $A^I \subseteq \Delta^I$; to every $r \in N_R$ a relation $r^I \subseteq \Delta^I \times \Delta^I$; and to every $a \in N_O$ an element $a^I \in \Delta^I$. The function $.^I$ extends to concepts as depicted in Table 1.

An interpretation I satisfies an axiom $\alpha (I \vDash \alpha)$ iff all I axioms and α are satisfied, i.e., I satisfies $C \sqsubseteq D$ iff $C^I \subseteq D^I, C(a)$ iff $a^I \in C^I, r(a, b)$ iff $\langle a, b \rangle \in r^I, r \subseteq s$ iff $r^I \subseteq s^I$. O entails $\alpha (O \vDash \alpha)$ iff every model of O is also a model of α. In this paper, variables are denoted by *x, y, z,* possibly with subscripts. Terms are variables or individuals.

4 Normal Form and Matrix Representation for $\mathcal{ALCHQ}^{=}$

Matrices with (qualified) number restrictions can be represented in two ways: the *abridged form*, i.e., with the number restrictions explicit, and the *expanded form*, with number restrictions substituted by axioms containing concepts, roles and (in)equalities that correspond to the semantic definitions. Besides, to take on (in)equalities, *substitutivity axioms* (e.g., $\forall x \forall y (x = y) \rightarrow (E(x) \rightarrow E(y))$ for concept names, and $\forall x \forall y \forall z \forall k (x = z) \wedge (y = k) \rightarrow (r(x, y) \rightarrow r(z, k))$ for role names) are represented as clauses for every concept and role name in the query.

Table 1. Syntax and semantics of $\mathcal{ALCHQ}^{=}$ constructors

Construct	Syntax	Semantics
Atomic negation	$\neg C$	$\Delta^I \setminus C^I$
Conjunction	$C \sqcap D$	$C^I \cap D^I$
Disjunction	$C \sqcup D$	$C^I \cup D^I$
Exist. restriction	$\exists r.C$	$\{x \in \Delta^I \mid \langle x, y\rangle \in r^I \wedge y \in C^I\}$
Value restriction	$\forall r.C$	$\{x \in \Delta^I \mid \langle x, y\rangle \in r^I \rightarrow y \in C^I\}$
(In)equality	$a = b \setminus \neq$	$a^I = b^I \setminus a^I \neq b^I$
Qualified number restriction[a] (for simple number restrictions, drop $\wedge y_i \in C^I$ from the semantics)	$\leq nr.C$	$\{x \in \Delta^I \mid \bigwedge_{i=1}^{n+1} \langle x, y_i\rangle \in r^I \wedge y_i \in C^I \bigwedge_{i,j=1, i\neq j}^{n} y_i \neq y_j \bigwedge_{i=1}^{n-1} y_i \neq y_{n+1} \rightarrow y_n = y_{n+1}\}$
	$\geq nr.C$	$\{x \in \Delta^I \mid \bigwedge_{i=1}^{n+1} \langle x, y_i\rangle \in r^I \wedge y_i \in C^I \bigwedge_{i,j=1, i\neq j}^{n+1} y_i \neq y_j\}$

[a]Note that we have relied on an unusual semantics for number restrictions, instead of $\{x \in \Delta^I \mid \#\langle x, y_i\rangle \in r^I \wedge y_i \in C^I \leq | \geq n\}$. The semantics presented here indeed consists of the basis for the number restrictions rules ($\leq | \geq$-rules [1]) in tableaux calculi.

Next, the matrix is converted to a specific DNF, introduced here. This DNF, with definitions concerning representation as matrices for the calculus, is presented below.

Definition 1 ($\mathcal{ALCHQ}^{=}$ literal, formula, clause, matrix). *$\mathcal{ALCHQ}^{=}$ literals* are atomic concepts or roles, possibly negated and/or instantiated, or *(in)equalities*. Literals involved in universal or existential restrictions are underlined. In case a restriction involves more than one clause, literals are indexed (in the top of the literal) with a same new column index number. An $\mathcal{ALCHQ}^{=}$ formula in DNF is a disjunction of conjunctions (like $C_1 \vee \ldots \vee C_n$), where each C_i has the form $L_1 \wedge \ldots \wedge L_m$, with each L_i being a literal. The *matrix* of an $\mathcal{ALCHQ}^{=}$ formula in DNF is a set $\{C_1, \ldots, C_n\}$, where each *clause* C_i has the form $\{L_1, \ldots, L_m\}$ with literals L_i.

Definition 2 (Substitutivity clauses, graphical matrix). $\mathcal{ALCHQ}^{=}$ matrices representing number restrictions also contain *substitutivity clauses* for every concept and role name, in the forms $\{x \neq y, E(x), \neg E(y)\}$ and $\{x \neq z, y \neq k, r(x, y), \neg r(z, k)\}$ with $E \in N_C, r \in N_R$.

In the *graphical matrix* representation, clauses are represented as columns, and restrictions as lines; restrictions with indexes are horizontal; without are vertical (see Example 3 – substitutivity axioms are not presented). Literals participating in a universal restriction in an axiom's left-hand side (LHS) or in an existential restriction in the right-hand side (RHS) are underlined; otherwise, they are sidelined.

Example 3 (Query, clause, $\mathcal{ALCHQ}^{=}$ matrix, abridged/expanded forms). Figure 3 shows query $O = \{ > 1\,\text{hasPart.Wheel} \sqsubseteq \text{Vehicle},\ \text{Car} \sqsubseteq \geq 3\,\text{hasPart.Wheel}\}$, $\alpha = \text{Car} \sqsubseteq \text{Vehicle}$ in abridged form. The index marks clauses involved in a same restriction).

$\{\{\underline{\geq 1\text{hasPart, Wheel}}, \neg\text{Vehicle}\}, \{\text{Car}, \underline{\leq 3\ \neg\text{hasPart}^1}\}, \{\text{Car}, \underline{\neg\text{Wheel}^1}\}, \{\text{Vehicle(a)}\}, \{\neg\text{Car(a)}\}\}$

$$\left[\begin{array}{l|llll} > 1\ \text{hasPart} & \text{Car} & \text{Car} & \neg\text{Car(a)} & \text{Vehicle(a)} \\ \text{Wheel} & \underline{< 3\ \neg\text{hasPart}} & \underline{\neg\text{Wheel}} & & \\ \neg\text{Vehicle} & & & & \end{array}\right]$$

Fig. 3. The query from Example 1 represented as an $\mathcal{ALCHQ}^{=}$ matrix in abridged form

The negations in literals $\underline{<3\neg\text{hasPart}^1}$ and $\underline{\neg\text{Wheel}^1}$ constitute merely a notational convention that facilitates the connections. They reflect the transformation to the expanded form, where these literals are converted into negated literals and equalities.

The number restriction expanded form, according to the semantics defined in Table 1, replaces $\underline{> 1\text{hasPart, Wheel}}$ by $\text{hasPart}(x, y_1) \sqcap \text{Wheel}(y_1) \sqcap \text{hasPart}(x, y_2) \sqcap \text{Wheel}(y_2) \wedge y_1 \neq y_2$ and $\underline{<3\ \neg\text{hasPart}^1}$ by $\bigwedge_{i=1}^{3} \text{hasPart}(x, v_i) \sqcap \text{Wheel}(v_i) \sqcap v_1 \neq v_2 \sqcap v_1 \neq v_3 \rightarrow v_2 = v_3$ before creating the matrix. The resulting matrix is depicted in Fig. 4. For the sake of space, substitutivity axioms are not shown.

$\{\{\underline{\text{hasPart, Wheel}(y_1)}, \underline{\text{hasPart, Wheel}(y_2)}, y_1 \neq y_2, \neg\text{Vehicle}\}, \{\text{Car}, \underline{\neg\text{hasPart}^1}\}, \{\text{Car}, \underline{\neg\text{Wheel}(v_1)^1}\}, \{\text{Car}, \underline{\neg\ \text{hasPart}^2}\}, \{\text{Car}, \neg\text{Wheel}(v_2)^2\}, \{\text{Car}, \underline{\neg\ \text{hasPart}^3}\}, \{\text{Car}, \underline{\neg\text{Wheel}(v_3)^3}\}, \{\text{Car}, v_1 = v_2\}, \{\text{Car}, v_1 = v_3\}, \{\text{Car}, v_2 = v_3\}, \{\text{Vehicle(a)}\}, \{\neg\text{Car(a)}\}\}$

$$\left[\begin{array}{l|llllllllll} \text{h} & \text{C} & \text{C} & \text{C} & \text{C} & \text{C} & \text{C} & \text{C} & \text{C} & \neg\text{C(a)} & \text{V(a)} \\ \text{W}(y_1) & \underline{\neg\text{h}} & \underline{\neg\text{W}(v_1)} & \underline{\neg\text{h}} & \underline{\neg\text{W}(v_2)} & \underline{\neg\text{h}} & \underline{\neg\text{W}(v_3)} & v_1 = v_2 & v_1 = v_3 & v_2 = v_3 & \\ \text{h} & & & & & & & & & & \\ \text{W}(y_2) & & & & & & & & & & \\ y_1 \neq y_2 & & & & & & & & & & \\ \neg\text{V} & & & & & & & & & & \end{array}\right]$$

Fig. 4. Same example in expanded form, showing the (in)equalities (again, literals are abridged, i.e., C means Car, h means hasPart, etc.)

$$i)\ 1NF\text{:}\ \left[\begin{array}{c} E_1 \\ \vdots \\ E_n \\ \neg D_1 \\ \vdots \\ \neg D_m \end{array}\right] \qquad ii)\ 2NF\text{:}\ \left[\begin{array}{cccc} E & \cdots & \cdots & E \\ \underline{\neg r} & \underline{\neg D_1} & \underline{\cdots} & \underline{\neg D_n} \end{array}\right] \qquad iii)\ 3NF\text{:}\ \left[\begin{array}{cccc} \underline{\neg r} & \underline{D_1} & \underline{\cdots} & \underline{D_m} \\ \neg E & \cdots & \cdots & \neg E \end{array}\right]$$

Fig. 5. Examples of the three two-lined normal forms' representations in $\mathcal{ALCHQ}^{=}$

Definition 3 (Impurity, pure conjunction/disjunction). *Impurity* in an $\mathcal{ALCHQ}^{=}$ formula is a disjunction in a conjunction, or a conjunction in a disjunction. A *pure conjunction* (PC) or *disjunction* (PD) does not contain impurities (see Definition in [6]).

Example 4 (Impurity, pure conjunction/disjunction). (a) $\exists r.A$ and $\bigwedge_{i=1}^{n} A_i$ are PCs if A and each A_i are also PCs; (b) $(\forall r.(D_0 \sqcup \ldots \sqcup D_n \sqcup (E_0 \sqcap \ldots \sqcap E_m) \sqcup (A_0 \sqcap \ldots \sqcap A_p))$ is not a PD, as it contains two impurities: $(E_0 \sqcap \ldots \sqcap E_m)$ and $(A_0 \sqcap \ldots \sqcap A_p)$.

Definition 4 (Two-lined disjunctive normal form). An $\mathcal{ALCHQ}^{=}$ axiom is in *two-lined DNF* iff it is in DNF and in one of the following normal forms (NFs): (i) $\widehat{E} \sqsubseteq \check{D}$; (ii) $E \sqsubseteq \widehat{E}$; (iii) $\check{D} \sqsubseteq E$, where E is a concept name[1] $\widehat{E}$ is a PC, and $\check{D}$ is a PD.

Example 5 (Two-lined disjunctive normal form). The axioms (i) $\widehat{E} \sqsubseteq \check{D}$; (ii) $E \sqsubseteq \exists r.\widehat{E}$ and (iii) $\forall r.\check{D} \sqsubseteq E$, where $\widehat{E} = \bigwedge_{i=1}^{n} E_i$ and $\check{D} = \bigvee_{j=1}^{m} D_j$ (Fig. 5).

Definition 5 (Cycle, cyclic/acyclic ontologies and matrices). If A and B are atomic concepts in an ontology O, A *directly uses* B, if B appears in the right-hand side of a subsumption axiom whose left-hand side is A. Let the relation *uses* be the transitive closure of *directly uses*. A *cyclic ontology* or *matrix* has a cycle when an atomic concept *uses* itself; otherwise it is *acyclic* [1]; e.g., $O = \{A \sqsubseteq \exists r.B, B \sqsubseteq \exists s.A\}$ is cyclic.

5 The $\mathcal{ALCHQ}^{=}$ θ-Connection Calculus ($\mathcal{ALCHQ}^{=}$ θ-CM)

The $\mathcal{ALCHQ}^{=}$ θ-Connection Method (henceforth $\mathcal{ALCHQ}^{=}$ θ-CM) differs from the FOL Connection Method (CM) by replacing Skolem functions and unification by θ-substitutions, and, just as typical DL systems, employs blocking to assure termination.

Besides, equality connections, proposed by Bibel [4], are needed here as a first attempt to address (in)equalities, and thus (qualified) cardinality restrictions. The idea is to include substitutivity axioms for each concept and role name, e.g., for concept P: $x = y \rightarrow (P(x), \rightarrow P(y))$, represented as a single column $\{x = y, P(x), \neg P(y)\}$

Moreover, w.r.t. $\mathcal{ALC}$ θ-CM, $\mathcal{ALCHQ}^{=}$ θ-CM expands the notion of connection to include equality, which is used to express number restrictions. An ontology represented as a matrix with the equalities is said to be in the *expanded form* and is explained in the next section. The *abridged form*, with number restrictions without equalities, is tackled in Subsect. 4.2.

5.1 Expanded Form - Representation and Reasoning

Definition 6 (Path, connection, θ-substitution, θ-complementary connection). A *path through a matrix M* contains exactly one literal from each clause/column in M. A *connection* is a pair of literals in three forms: (i) $\{E, \neg E\}$ with the same concept/role name, instantiated with the same instance(s) or not; (ii) $\{x = y, x \neq y\}$, with x and y instantiated with the same instance or not. A θ-*substitution* assigns each (possibly omitted) variable an individual or another variable, in an $\mathcal{ALCHQ}^{=}$ literal. A θ-*complementary* connection is a

[1] The symbols E and $\widehat{E}$ were chosen here to designate a concept name and a pure conjunction rather than the usual C and $\widehat{C}$, to avoid confusion with clauses, that are also denoted by C.

pair of $\mathcal{ALCHQ}^{=}$ literals $\{E(x), \neg E(y)\}$ or $\{p(x,v), \neg p(y,u)\}$, with $\theta(x) = \theta(y)$, $\theta(v) = \theta(u)$. The complement $\overline{L}$ of a literal L is E if $= \neg E$, and it is $\neg E$ if $L = E$.

Remark 1 (θ-substitution). Simple term unification without Skolem functions is used to calculate θ-substitutions. The application of a θ-substitution to a literal is an application to its variables, i.e. $\theta(E) = E(\theta(x)), x$ fresh, and $\theta(r) = r(\theta(x), \theta(y))$, where E is an atomic concept and r is a role. For notation, $x^{\theta} = \theta(x)$.

Definition 7 (Set of concepts). The *set of concepts* $\tau(x)$ of a term x contains all concept names instantiated by x so far, defined as $\tau(x) \underline{\underline{\text{def}}} \{E \in N_C | E(x) \in Path\}$.

Definition 8 (Skolem condition). The *Skolem condition* ensures that at most one concept name is underlined for each term in the graphical matrix form. If i is an index, this condition is defined as $\forall a \mid \{\underline{E^i} \in N_C \mid \underline{E^i(a)} \in Path\} \mid \leq 1$.

Definition 9 ($\mathcal{ALCHQ}^{=}$ connection calculus). Figure 6 brings the formal $\mathcal{ALCHQ}^{=}$ connection calculus ($\mathcal{ALCHQ}^{=}$ θ-CM), adapted from the FOL CM [11]. The rules of the calculus are applied in an analytic, bottom-up way. The basic structure is the tuple <*C, M, Path*> , where clause C is the open sub-goal, M the matrix corresponding to the query $O \models \alpha$ (O is an $\mathcal{ALCHQ}^{=}$ ontology) and *Path* is the *active path*, i.e. the (sub-)path being currently checked. The index $\mu \in \mathbb{N}$ of a clause C^{μ} denotes that C^{μ} is the μ-th copy of clause C, increased when *Cop* is applied for that clause (the variable x in C^{μ} is denoted x_{μ}) – see example of copied clauses in Fig. 13*a*. When *Cop* is applied, it is followed by the application of *Ext* or *Red*, to avoid non-determinism in the rules' application. The *Blocking Condition* states that, when a cycle finishes, the last new individual x_{μ}^{θ} (if it is new, then $x_{\mu}^{\theta} \notin N_O$, as in the condition) has a set of concepts $\tau(x_{\mu}^{\theta})$ which is not a subset of the set of concepts of the previous copied individual, i.e., $\tau(x_{\mu}^{\theta}) \not\subseteq \tau\left(x_{\mu-1}^{\theta}\right)$ [14]. If this condition is not satisfied, blocking occurs.

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

$$Start\ Rule\ (St) \quad \frac{C_1, M, \{\}}{\varepsilon, M, \varepsilon} \quad with\ C_1 \in \alpha$$

$$Reduction\ Rule\ (Red) \quad \frac{C, M, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}}$$

$$with\ \theta(L_1) = \theta(\overline{L_2})\ \ and\ the\ Skolem\ condition\ holds$$

$$Extension\ Rule\ (Ext) \quad \frac{C_1 \backslash \{L_2\}, M, Path \cup \{L_1\} \qquad C, M, Path}{C \cup \{L_1\}, M, Path}$$

$$with\ C_1 \in M,\ L_2 \in C_1, \theta(L_1) = \theta(\overline{L_2})\ \ and\ the\ Skolem\ condition\ holds$$

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

$$with\ C_2^{\mu}\ is\ a\ copy\ of\ C_1, L_2 \in C_2^{\mu}, \theta(L_1) = \theta(\overline{L_2})\ and\ the\ blocking\ condition\ holds$$

Fig. 6. The connection calculus $\mathcal{ALCHQ}^{=}$ θ-CM

Lemma 1 (Matrix characterization). A matrix M is valid iff there exist an index μ, a set of θ-substitutions $\langle\theta_i\rangle$ and a set of connections S, s.t. every path through M^{μ}, the matrix with copied clauses, contains a θ-complementary connection $\{L_1^{\theta}, L_2^{\theta}\} \in S$, i.e. a connection with $\theta(L_1) = \theta(\overline{L_2})$. The tuple $\langle\mu, \langle\theta_i\rangle, S\rangle$ is called a *matrix proof*.

Clause copying and its multiplicity μ already existed in the original CM, but neither a copy rule nor blocking were necessary, as FOL is semi-decidable. To regain termination, the new *Copy* rule implements blocking [1], when no alternative connection is available and cyclic ontologies are being processed. The rule regulates the creation of new individuals, blocking when infinite cycles are detected. The *Skolem condition* solves the FOL cases where the combination of Skolemization and unification correctly prevents connections (see Soundness Theorem in WWW.cin.ufpe.br/∼fred/RR.pdf).

In the *Ext* and *Red* rules, θ-substitutions replace implicit variables by terms in the current path. A restriction avoids the situation in FOL matrices, where unification is tried with distinct Skolem functions: any individual x can have in its set of concepts $\tau(x)$ at most a single concept name with a column index in the matrix, stated by the condition $\forall a \left| \left\{ \underline{E^i} \in N_C \middle| \underline{E^i(a)} \in Path \right\} \right| \leq 1$.

Example 6 ($\mathcal{ALCHQ}^{=}$ connection calculus). Figures 7 and 8 show the proof of the query from Example 1 using the matrix representation and the calculus, respectively.

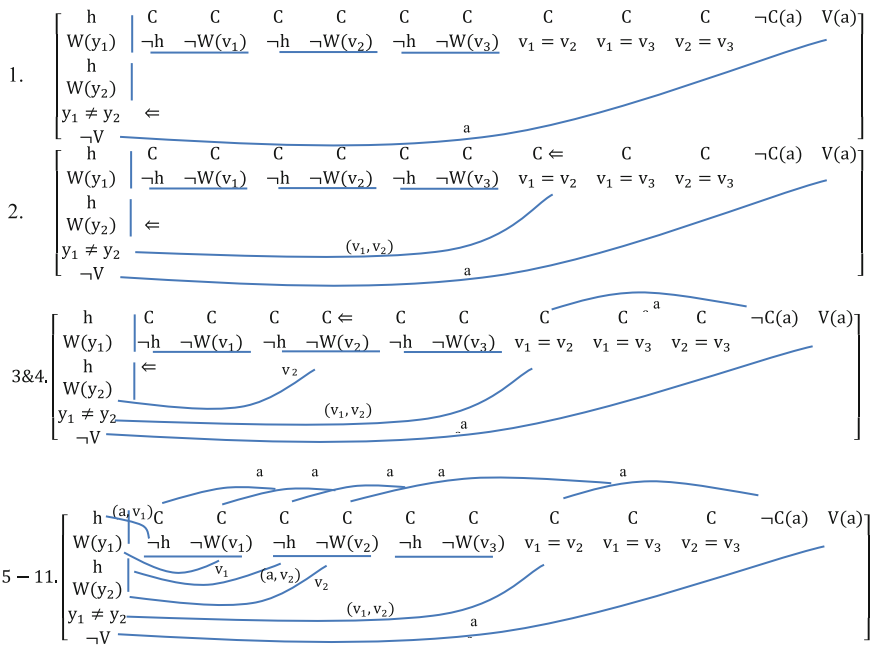

Fig. 7. The query's proof in graphical matrix representation. Arcs are connections whose labels are the names of the involved individual(s)/variable(s). Arrows indicate pending literals' lists.

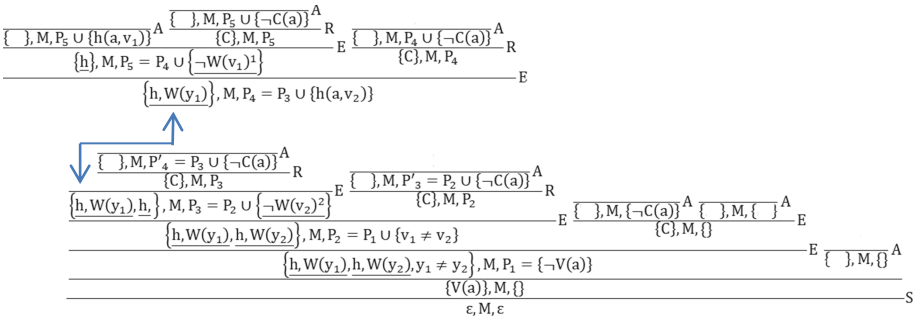

Fig. 8. The proof of the query using the calculus, where M is an abbreviation for $\{\{\underline{h, W(y_1)}, \underline{h, W(y_2)}, y_1 \neq y_2, \neg V\}, \{C, \underline{\neg h^1}\}\}$, $\{C, \underline{\neg W(v_1)^1}\}, \{C, \underline{\neg h^2}\}\}, \{C, \underline{\neg W(v_2)^2}\}, \{C, \underline{\neg h^3}\}\}$, $\{C, \underline{\neg W(v_3)^3}\}$, $\{C, v_1 = v_2\}$, $\{C, v_1 = v_3\}$, $\{C, v_2 = v_3\}$, $\{V(a)\}$, $\{\neg C(a)\}\}$. The double-ended arrow just copies the proof part to save text space.

Furthermore, when equality between pairs of individuals are being dealt, equality connections [4] with substitutivity axioms, in explicit or implicit form, can be relied upon. One can solve, e.g., $\{P(a), a = b\} \models P(b)$, as portrayed in Fig. 9. Figure 9(i) displays the equality connections performed in the usual way, with the introduction of the substitutivity axiom P: $x = y \rightarrow (P(x) \rightarrow P(y))$ (represented as the column $\{x = y, P(x), \neg P(y)\}$), while Fig. 9(ii) presents the same connection in an abridged way.

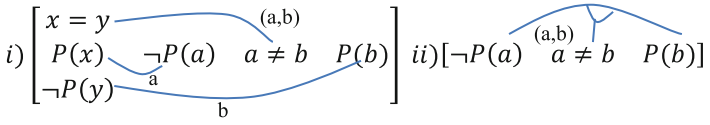

Fig. 9. (*i*) A connection using the substitutivity axiom; (*ii*) an equality connection [4]

This subject naturally leads to the representation of number restrictions connections in the abridged form, deployed in the next subsection.

5.2 Abridged Form - Representation and Reasoning

(Qualified) number restrictions can be in *abridged form* ($\geq \mid \leq nr(.C)$ with $n \in \mathbb{N}^*$). In this case, one should note that $\neg(\geq nr) = \leq (n-1)r$ and $\neg(\leq nr) = \geq (n+1)r$.

Definition 10 (Number restriction literal). *Number restriction literals* are literals representing (qualified) number restrictions. They can be negated and/or instantiated, and/or under- or sidelined or with no line. In case a restriction involves more than one clause, literals are top indexed with a same new column index number.

Definition 11 (Number restriction valid interval). Two number restrictions form a *valid interval* iff their numerical restrictions share an intersection, e.g. $>5r$, $<8\ \neg r$.

Definition 12 (Number restriction θ-substitution, θ-complementary number restriction connection). Let A and B be two number restriction literals, $\leq | \geq nr$ and $\geq | \leq m\ \neg r$, instantiated or not, representing role instance sets $\langle r(x, y_1), \ldots, r(x, y_n)\rangle$ and $\langle r(z, w_1), \ldots, r(z, w_m)\rangle$, with a valid interval between them (vi). A *number restriction θ-substitution* for the pair is a mapping θ, s.t. $\theta(x) = \theta(y), \theta(y_i) = \theta(w_i)$, with $i = 1$ to $\min(vi)$. A *θ-complementary number restriction connection* is a pair of number restriction literals over a same role in the form $\{\leq | \geq n\, r, \geq | \leq m\ \neg r\}$, that, under a number restriction θ-substitution, share a valid interval vi.

A connection represents a tautology, e.g. $E \sqcup \neg E$. For number restrictions, this means a valid interval, as, for example, any individual possessing any number of role instances (including 0) with r satisfies the restriction $>5r \sqcup <8r$. If there is a "hole", for instance, $>8r \sqcup <5r$, then individuals with 5 to 8 role instances of r would not satisfy the restriction, and the latter cannot be a tautology. Recall that $<8r$ is represented as $<8\ \neg r$, only to facilitate the connections to be settled.

Example 6 ($\mathcal{ALCHQ}^{=}$ connection calculus, abridged form). Figures 10 and 11 display the proof from Example 2 in the abridged form, using the graphical matrix representation and the formal calculus. Note that $\min(> 1\,\text{hasPart}, < 3\ \neg\text{hasPart}) = 2$.

The abridged form can easily accommodate number restrictions with role hierarchies, if connections between number restrictions and role axioms exist.

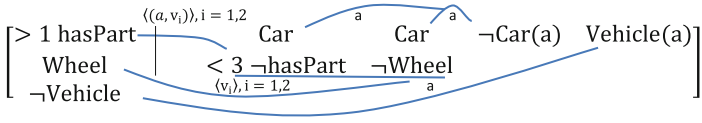

Fig. 10. Proof of Example 2 in the abridged form. $\langle(a, \mathrm{v_i})\rangle$, $\mathrm{i} = 1, 2$ is a set of two role instances and $\mathrm{v_i}, \mathrm{i} = 1, 2$ is a set of two instances (of concept Wheel).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{\},\mathrm{M},\mathrm{P_3} = \mathrm{P_2} \cup \{\underline{\neg \mathrm{W}\langle \mathrm{v_i}\rangle^1}\},\ \mathrm{i} = 1,2}\,\mathrm{A} \quad \dfrac{\dfrac{}{\{\},\mathrm{M},\mathrm{P'_3} = \mathrm{P_2} \cup \{\neg \mathrm{C(a)}\}}\,\mathrm{A}}{\{\mathrm{C}\},\mathrm{M},\mathrm{P_2}}\,\mathrm{R}}{\{\mathrm{W}\langle \mathrm{v_i}\rangle\},\mathrm{M},\mathrm{P_2} = \mathrm{P_1} \cup \{\underline{\neg \mathrm{h}\langle(\mathrm{a}, \mathrm{v_i})\rangle^1}\},\ \mathrm{i} = 1,2}\,\mathrm{E} \quad \dfrac{\dfrac{}{\{\},\mathrm{M},\{\neg \mathrm{C(a)}\}}\,\mathrm{A} \quad \dfrac{}{\{\},\mathrm{M},\{\}}\,\mathrm{A}}{\{\mathrm{C}\},\mathrm{M},\{\neg \mathrm{V(a)}\}}\,\mathrm{E}}{\{\underline{> 1\,\mathrm{h}, \mathrm{W}}\},\mathrm{M},\mathrm{P_1} = \{\neg \mathrm{V(a)}\}}\,\mathrm{E} \quad \dfrac{}{\{\},\mathrm{M},\{\}}\,\mathrm{A}}{\{\mathrm{V(a)}\},\mathrm{M},\{\}}$$
$$\varepsilon,\mathrm{M},\varepsilon\ \mathrm{S}$$

Fig. 11. The proof of Example 2 using the calculus, with $M = \{\{> 1\text{hasPart}, \text{Wheel}, \neg\text{Vehicle}\}, \{\text{Car}, \underline{< 3\ \neg\text{hasPart}^1}\}, \{\text{Car}, \underline{\neg\text{Wheel}^1}\}, \{\text{Vehicle(a)}\}, \{\neg\text{Car(a)}\}\}$ (literals are abbreviated)

Example 7 (Number restrictions, role hierarchies). $O = \{ > 2\,\text{hasPart.Wheel} \sqsubseteq \text{Car}, \text{hasComponent} \subseteq \text{hasPart}, \text{Truck} \sqsubseteq \geq 6\,\text{hasPart.Wheel}\}$, $\alpha = \text{Truck} \sqsubseteq \text{Car}$.

This query is represented by $M = \{\{ > 2\ \text{hasPart}, \text{Wheel}, \neg\text{Car}\}, \{\text{hasComponent}\ , \neg\text{hasPart}\}, \left\{\text{Truck}, \underline{< 5\ \neg\text{hasComponent}^1}\right\}, \{\text{Truck}, \underline{\neg\text{Wheel}^1}\}, \{\neg\text{Truck(a)}\}, \{\text{Car(a)}\}\,.$

Figure 12 brings the proof for M, with $\min(> 2\,\text{hasPart}, < 5\,\text{hasPart}) = 3$.

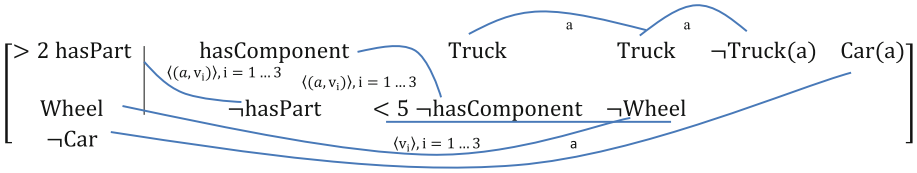

Fig. 12. Proof with number restrictions and a role hierarchy axiom

6 Discussion

Matricial inference methods, such as the CM, presents a few advantages over other methods, as well as some drawbacks. We will discuss our method, at first in the light of memory handling and existent solutions to solve equality equations in the context of FOL. Next, we briefly comment some recent comparative performance of our $\mathcal{ALC}$ reasoner, RACCOON (ReAsoner based on the Connection Calculus Over ONtologies) against well-known DL reasoners [7], and existent solutions for number restrictions within the DL scenario, followed by a small discussion on next steps.

As for memory usage, in the CM, matrices require only a copy of the matrix and data structures to store the current path, the pending clauses and literals, the unifier and literal's indices. It does not generate any intermediary results; this constitutes an interesting benefit in terms of memory usage over *generative* methods such as resolution or tableaux, which create intermediary clauses and sub-formulae.

Indeed, dealing efficiently with memory with cyclic ontologies is crucial for a DL reasoner, since a number of fragments (including $\mathcal{ALCN}$-Aboxes) have been proven PSPACE-complete [1]. Our calculus processes cycles (thanks to the *Copy rule*), saving memory due to keeping only one copy of the matrix in memory [3, 4]. The other copies are virtual, i.e., only the index μ is created or incremented and stored, together with the θ-substitution and the current path. The next example portraits this case.

Example 8 (Cycles). $O = \{\exists\text{hasSon}.\ (\text{Dr} \sqcup \text{DrAncestor}) \sqsubseteq \text{DrAncestor}, \text{hasSon}\ (\text{ZePadre},\ \text{Moises}), \text{hasSon}\ (\text{Moises}, \text{Luiz}), \text{hasSon}\ (\text{Luiz},\ \text{Fred}),\ \text{Dr}\ (\text{Fred})\},\ \alpha = \text{DrAncestor}(\text{ZePadre})$. This cyclic query has its proof represented by both Figs. 13*a* and *b*.

Figure 13*a* brings an explicit copy of the second clause, needed for the proof. On the other hand, Fig. 13*b* incorporate indices to denote how the only copy was used with different individuals and instantiations. At least in theory, such idea exists in the CM,

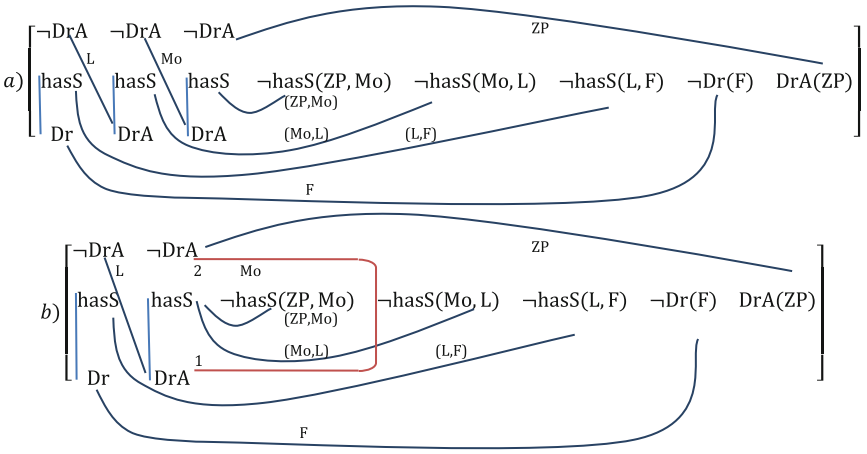

Fig. 13. Proof representations of a cyclic query, with (a) explicit and (b) implicit copies

called *implicit amplification* [3]; we adopted it in RACCOON with the same notation, and gain memory with its procedure.

Clausal inference methods require normal forms, in which transformations apply over formulae to produce clauses over which the method works. On the one hand, clause manipulation accelerates reasoning in reasonably expressive logics, e.g., FOL. On the other hand, the drawbacks are at least two-fold.

First, literals' redundancy among clauses often constitutes an overhead in large knowledge bases. In the CM, matrix representation minors the problem during reasoning, as the method is non-generative; anyway, it remains if, in an initial query representation in DNF, clauses share too many literals. For the $\mathcal{ALCHQ}^{=}$ θ-CM, the two-lined normal form reduces this type of redundancy at the expense of introducing a small number of new symbols. To sum up, the best solution consists in applying a non-clausal connection method [12], where matrices can be nested.

Another problem for clausal calculi resides on adapting to an increasing set of constructs in DL: each new construct to be inserted into the calculi requires careful analysis, and frequently changes in the existing rules. This problem also plagues equality approaches in clausal systems. Consolidated solutions from saturation-based reasoning, such as paramodulation [13], are hard to be integrated, and the former is not complete for the connection method [11]. Nevertheless, an equality approach based on RUE (Resolution with Unification and Equality) [5] seems plausible for connection calculi but has not been tried yet. Our aim in formalizing our calculus with the Eq system is paving the way for such more efficient solutions.

Although the Eq system is not yet coded in RACCOON, the goal-oriented search embodied by the connection calculus, together with its economical approach to memory, made the reasoner display unexpected fair results for $\mathcal{ALC}$ consistency, compared to Hermit, FacT++ and Konclude. A summary of the benchmarking conducted over the ORE 2014 and 2015 baselines is deployed in Fig. 14 [7].

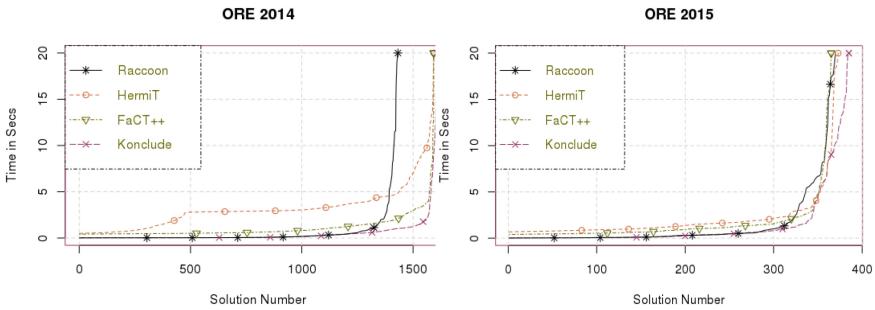

Fig. 14. Comparison of RACCOON and ORE competitors for consistency on the ORE 2014 and 2015 baselines ($\mathcal{AL}$, $\mathcal{ALE}$ and $\mathcal{ALC}$ ontologies)

In the baselines, ontologies were ranked by size and expressivity. RACCOON exhibited the fastest results (side by side with Konclude) in smaller and less complex ontologies; however, against the larger and more complex set (the last ones), results start to decay (in a graceful fashion), probably due to the lack of DL optimizations. Furthermore, in the first experiment, RACCOON's performance fell short in ontologies in the presence of a certain structure where cycles occur inside other cycles massively. Apart from that, the results seem promising, given the possibility of implementing reductions built in other competitors.

When faced with number restrictions and their equalities, the idea is applying the abridged form first, which demand less steps and memory; only in the cases it does not suffice, the expanded form must be used (comparing two number restrictions has a quadratic complexity in the simpler cases, not to talk about checking the ABox). Besides, with the expanded form, hundreds of substitutivity axioms might need to be added to the matrix. Thus, $\mathcal{ALCHQ}^{=}$ θ-CM can only be competitive in this DL fragment, when, e.g., solutions based on rewriting [2, 10] can be devised and integrated, i.e., a way to substitute equal individuals by their canonical representative is envisaged. Bibel already suggested term rewriting as a possible technique to solve equality in the CM [4]. Integrating it with the $\mathcal{ALCHQ}^{=}$ θ-CM represents a challenge for our calculus to remain competitive as more expressive fragments are to be addressed.

7 Conclusions and Future Work

In the current work, $\mathcal{ALCHQ}^{=}$ $\theta-$CM is presented, a connection method that enhances the $\mathcal{ALC}$ $\theta-$CM, by, mainly, introducing (in)equalities and, as a respective solution to handle them, equality connections with equality predicate substitutivity axioms explicit or implicit, as defined by Bibel. Two new forms of representing number restrictions are also shown: the abridged and the expanded form. In the former, cardinality restrictions are a new type of literals themselves, and this new notion of literal together with its respective new connection type had to be defined. In the latter, number restrictions are replaced by literals and (in)equalities that correspond to the number restriction's semantic definition.

As for theoretical future work, we aim to create more sophisticated blocking schemes for dynamic and double blocking for DL constructs like inverses, union, intersection and complement of roles [9], transitivity, role chains and value maps, complex role axioms and dealing with nominals. As for practical future work, we intend to enhance the fragment currently dealt by RACCOON to include $\mathcal{ALCHQ}^{=}$, as well as the future new solutions mentioned as theoretical future work.

Acknowledgements. This work was partially supported by the project Reconciling Description Logics and Non-Monotonic Reasoning in the Legal Domain (PRC CNRS–FACEPE France–Brazil) and the anonymous reviewers. Fred Freitas also thanks Jens Otten, Evandro and Patty Travassos, for the personal support.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
2. Bate, A., Motik, B., Cuenca Grau, B., Simancik, F., Horrocks, I.: Extending consequence-based reasoning to $\mathcal{SHIQ}$. In: Workshop on Description Logics (DL), CEUR, pp. 34–46 (2015)
3. Bibel, W.: Matings in matrices. Commun. ACM **26**, 844–852 (1983)
4. Bibel, W.: Deduction – Automated Logic. Academic Press, London (1993)
5. Digricoli, V., Harrison, M.: Equality-based Binary Resolution. J. ACM **33**(2), 253–289 (1986)
6. Freitas, F.: A connection calculus over the description logic $\mathcal{ALC}$. In: Canadian Conference on Artificial Intelligence (AI), Victoria, Canada (2016)
7. Freitas, F., Melo, D., Otten, J.: RACCOON: a connection reasoner for $\mathcal{ALC}$. In: Proceedings of International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR) (2017)
8. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. J. Autom. Reason. **53**(3), 245–269 (2014)
9. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. J. Logic Comput. **9**(3), 385–410 (1999)
10. Motik, B., Nenov, Y., Piro, R., Horrocks, I.: Combining rewriting and incremental materialisation maintenance for datalog programs with equality. In: IJCAI, pp. 3127–3133 (2015)
11. Otten, J.: Restricting backtracking in connection calculi. AI Commun. **23**(2–3), 159–182 (2010)
12. Otten, J.: nanoCoP: natural non-clausal theorem proving. In: Proceedings of IJCAI, pp. 4924–4928 (2017)
13. Robinson, G., Wos, L.: Paramodulation and Theorem Proving in First-Order Theories with Equality. Mach. Intell. **4**, 135–150 (1969)
14. Schmidt, R., Tishkovsky, D.: Analysis of blocking mechanisms for description logics. In Proceedings of the Workshop on Automated Reasoning (2007)
15. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: system description. J. Web Semant. Sci. Serv. Agents World Wide Web **27**(1), 78–85 (2014)
16. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006). https://doi.org/10.1007/11814771_26