Algebraic Tableau Algorithm for $\mathcal{ALCOQ}$

Jocelyne Faddoul[1], Volker Haarslev[1], and Ralf Möller[2]

[1] Concordia University, Montreal, Canada
{j_faddou, haarslev}@encs.concordia.ca
[2] Hamburg University of Technology, Hamburg, Germany
r.f.moeller@tuhh.de

1 Introduction

Description Logics (DLs) are a family of knowledge representation formalisms used to represent and reason about an application's domain elements. They are applicable in the semantic web as they provide the basis for the Web Ontology Language (OWL).

Decision procedures for expressive DLs enabling both nominals and QCRs were published in [10] with very weak optimizations if any (no DL reasoner was able to classify the WINE[3] ontology until recent efforts [15]). Nominals are challenging in reasoning because in order to preserve their semantics, each *nominal* must be interpreted as exactly one individual, whereas a concept is interpreted as a set of individuals. Another challenge is that ontologies that use nominals no longer enjoy the quasi-tree model property which has always been advantageous for tableau. Major inefficiency sources can be due to (i) the high degree of non-determinism introduced by the use of GCIs or when merging domain elements is necessary, (ii) the construction of large models, or (iii) the interaction between constructors. The worst case occurs when *nominals* interact with *inverse roles* ($\mathcal{I}$) and *QCRs*. Each of these constructs alone is challenging to reason with and needs special optimization techniques. Resolution-based reasoning procedures were proposed in [18] without being well suited in dealing with large numbers in QCRs. Hypertableaux [13] were recently studied to minimize non-determinism in DL reasoning with no special treatment for QCRs. Combining tableau and algebraic reasoning dramatically improved performance with QCRs [5].

In this paper, we extend our previous work in [3] to additionally allow nominals and unfoldable TBoxes. With nominals, the handling of numerical restrictions becomes more challenging since numerical restrictions imposed by nominals interact with restrictions specified with QCRs. The algorithm is based on the assumption that domain elements consists of a set of individuals divided into subsets depending on their role filler membership and/or concept membership. Also nominals can be seen as singleton sets and the at-least and at-most restrictions expressed in QCRs represent cardinality restrictions on the corresponding sets of role fillers. These restrictions on sets are encoded as linear inequations solved by a linear programming (LP) algorithm (such as simplex) with the objective of minimizing the sum of all cardinalities. If no solution for the inequations is possible, this means that the individuals cannot be distributed between sets without violating the cardinality restrictions. When a solution is returned, a

[3] `http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf`

minimum number of individuals is distributed among sets without violating any at-least or at-most restrictions. We also introduce proxy individuals as representative of domain elements with common restrictions thus reducing the size of a completion graph model.

2 The Description Logic $\mathcal{ALCOQ}$

Let N_C, N_R be non-empty and disjoint sets of concept names and role names respectively. Let $N_o \subseteq N_C$ be the set of nominals. The set of $\mathcal{ALCOQ}$ concepts is the smallest set such that: (i) every concept name $A \in N_C$ is a concept, and (ii) if C, D are concepts and R is a role name in N_R then $\neg C$, $(C \sqcup D)$, $(C \sqcap D)$, $(\exists R.C)$, $(\forall R.C)$, $(\geq nR.C)$, $(\leq nR.C)$ with $n \in \mathbb{N}$ are also concepts. In the following we use $\top$ ($\bot$) as an abbreviation for $A \sqcup \neg A$ ($A \sqcap \neg A$) and $\geq nR$ ($\leq nR$) for $\geq nR.\top$ ($\leq nR.\top$). Also since $\exists R.C$ can be converted to $\geq 1R.C$, we do not consider input descriptions of the form $\exists R.C$.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of $\Delta^{\mathcal{I}}$, a non-empty set of individuals, called the domain of the interpretation, and $\cdot^{\mathcal{I}}$, an interpretation function. The interpretation function $\cdot^{\mathcal{I}}$ maps each atomic concept $A \in N_C$ to a subset of $\Delta^{\mathcal{I}}$, and each atomic role $R \in N_R$ to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Given role name $R \in N_R$ and an element $s \in \Delta^{\mathcal{I}}$, the set of R-fillers for s $FIL(R, s)$ is defined as $FIL(R, s) = \{t \in \Delta^{\mathcal{I}} \mid \langle s, t\rangle \in R^{\mathcal{I}}\}$ and the set of all *R-fillers* for R is defined as: $FIL(R) = \bigcup_{s \in \Delta^{\mathcal{I}}} FIL(R, s)$. We use # to denote the cardinality of a set and the semantics of all $\mathcal{ALCOQ}$ concepts satisfies the following:

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
\#o^{\mathcal{I}} &= 1 \text{ for all } o \in N_o \\
(\forall R.C)^{\mathcal{I}} &= \{s \in \Delta^{\mathcal{I}} \mid \langle s, t\rangle \in R^{\mathcal{I}} \Rightarrow t \in C^{\mathcal{I}}\} \\
(\geq nR.C)^{\mathcal{I}} &= \{s \in \Delta^{\mathcal{I}} \mid \#(FIL(R, s) \cap C^{\mathcal{I}}) \geq n\} \\
(\leq nR.C)^{\mathcal{I}} &= \{s \in \Delta^{\mathcal{I}} \mid \#(FIL(R, s) \cap C^{\mathcal{I}}) \leq n\}
\end{aligned}
$$

An unfoldable $\mathcal{ALCOQ}$ TBox $\mathcal{T}$ is a finite set of *concept subsumption axioms* of the form $A \sqsubseteq C$, and *concept definition axioms* of the form $A \equiv C$ (which abbreviates $\{A \sqsubseteq C,\ C \sqsubseteq A\}$) such that all axioms are unique and acyclic. $\mathcal{T}$ is said to be consistent if there exists an interpretation $\mathcal{I}$ satisfying $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each $C \sqsubseteq D \in \mathcal{T}$. $\mathcal{I}$ is called a model of $\mathcal{T}$. A concept C is said to be satisfiable w.r.t. an unfoldable TBox $\mathcal{T}$ iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$, i.e., there exists an individual $s \in C^{\mathcal{I}}$ as an instance of C. $\mathcal{I}$ is called a model of C w.r.t. $\mathcal{T}$.

An $\mathcal{ALCOQ}$ ABox $\mathcal{A}$ is a finite set of concept membership assertions of the form $a\!:\!C$ or role membership assertions of the form $(a, b)\!:\!R$ with a, b two individual names. $\mathcal{A}$ is said to be consistent w.r.t $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ is satisfied for each $a\!:\!C$ in $\mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ is also satisfied for each $(a, b)\!:\!R$ in $\mathcal{A}$.

Using nominals, concept satisfiability and ABox consistency can be reduced to TBox consistency; a concept C is satisfiable w.r.t. $\mathcal{T}$ iff $(\mathcal{T} \cup \{o \sqsubseteq C\})$ is consistent and $o \in N_o$ new in $\mathcal{T}$. For example, the concept description in (1) is satisfiable w.r.t the TBox defined in (2) iff the TBox in (3) is consistent. An ABox $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff $(\mathcal{T} \cup \bigcup_{(a:C) \in \mathcal{A}} \{a \sqsubseteq C\} \cup \bigcup_{((a,b):R) \in \mathcal{A}} \{a \sqsubseteq \exists R.b\})$ is consistent. Hence, in the following without loss of generality we restrict our attention to TBox consistency.

$$\geq 1S_1.(A \sqcap B_1) \sqcap \geq 1S_2.(A \sqcap B_2) \tag{1}$$

$$\mathcal{T} = \{A \sqsubseteq \geq 1R.B, B_1 \sqsubseteq \forall R.C, B_2 \sqsubseteq \forall R.\neg C\} \tag{2}$$

$$\mathcal{T} = \{A \sqsubseteq \geq 1R.B, B_1 \sqsubseteq \forall R.C, B_2 \sqsubseteq \forall R.\neg C, o \sqsubseteq \geq 1S_1.(A \sqcap B_1) \sqcap \geq 1S_2.(A \sqcap B_2)\} \tag{3}$$

$$\mathcal{T} = \{o \sqsubseteq \geq 1S_1.(\geq 1R.B \sqcap \forall R.C) \sqcap \geq 1S_2.(\geq 1R.B \sqcap \forall R.\neg C)\} \tag{4}$$

$$C_{\mathcal{T}} = (\neg o \sqcup \geq 1S_1.(\geq 1R.B \sqcap \forall R.C) \sqcap \geq 1S_2.(\geq 1R.B \sqcap \forall R.\neg C)) \tag{5}$$

In the following, we assume all concepts to be in their *negation normal form* (NNF). We use $\dot{\neg}C$ to denote the NNF of $\neg C$ and $nnf(C)$ to denote the NNF of C. The concept axioms in $\mathcal{T}$ can be reduced to a single axiom $\top \sqsubseteq C_{\mathcal{T}}$ such that $C_{\mathcal{T}}$ abbreviates $\bigsqcap_{C \sqsubseteq D \in \mathcal{T}} nnf(\neg C \sqcup D)$ [10] and $\mathcal{T}$ has been unfolded. $\mathcal{T}$ can be unfolded in the following way: (i) for every $A \equiv C \in \mathcal{T}$ then occurrences of A are replaced by C, (ii) for every $A \sqsubseteq C$ then occurrences of A such that A is not negated are replaced by C and otherwise $\neg A$ is left unchanged. Unfolding the TBox in (3) would result in the TBox in (4) which can be reduced to the single axiom in (5). A TBox consistency test can be checked by testing the consistency of $o \sqsubseteq C_{\mathcal{T}}$ with $o \in N_{\mathrm{o}}$ new in $\mathcal{T}$, which means that at least $o^{\mathcal{I}} \subseteq {C_{\mathcal{T}}}^{\mathcal{I}}$ and ${C_{\mathcal{T}}}^{\mathcal{I}} \neq \emptyset$. Moreover, since $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ then every domain element must also satisfy $C_{\mathcal{T}}$ ($C_{\mathcal{T}}$ must hold at each domain element).

3 Motivation

Nominals carry implicit numerical restrictions; they not only name individuals but also allow us to count them. This additional information carried with nominals interacts with concepts and roles in a way that can limit the number of individual (i) members of a certain concept C, or (ii) fillers for a certain role R. Given an individual s, instance of a concept E ($s \in E^{\mathcal{I}}$), $C \in N_{\mathrm{C}}$, $R \in N_{\mathrm{R}}$, $o_1, \ldots, o_n \in N_{\mathrm{o}}$, and n, m non-negative integers, we distinguish between and define *global* and *local* numerical restrictions.

Definition 1 (Global Restrictions). Having $E \sqsubseteq o_1 \sqcup \cdots \sqcup o_n \in \mathcal{T}$ ($o_1 \sqcup \cdots \sqcup o_n \sqsubseteq E \in \mathcal{T}$) enforces a numerical restriction on the cardinality of the set of instances of E; there can be at-most (at-least) n instances of E corresponding to the interpretation of $o_1^{\mathcal{I}} \cup \ldots \cup o_n^{\mathcal{I}}$ assuming $o_1, \ldots, o_n$ are mutually disjoint. Such at-most (at-least) restrictions carried with nominals are global since they can affect the set of all individuals in the domain of interpretation ($\Delta^{\mathcal{I}}$). Nominals can specify concept cardinalities [1, 16] and can interact with local restrictions. For example in (6) the concept BloodType has exactly 8 instances (assuming that the nominals $o^+, \ldots, AB^-$ are mutually disjoint) and having $s \in (\forall R.BloodType)^{\mathcal{I}}$ then the set of *R-fillers* of s is bounded by 8, $\#FIL(R, s) = 8$.

$$BloodType \equiv o^+ \sqcup A^+ \sqcup B^+ \sqcup AB^+ \sqcup o^- \sqcup A^- \sqcup B^- \sqcup AB^- \tag{6}$$

Definition 2 (Local Restriction). QCRs carry explicit numerical restrictions; when E is of the form ($\geq nR.C$), or ($\leq mR.C$) it holds a restriction on the cardinality of the set of *R-fillers* of s. These restrictions are local to $FIL(R, s)$. For example, having $s \in (Adult)^{\mathcal{I}}$ with Adult defined in (7) imposes that at least 306 individuals s_1, s_2, …, s_{306} must be *hasBone-filler*s of s, and therefore $\#FIL(R, s) \geq 306$.

$$Adult \sqsubseteq \geq 306\, hasBone.Bone \tag{7}$$

The interaction between *global* and *local restrictions* makes the problem of reasoning with nominals twofold. First, nominals break the advantageous tree model property of TBoxes and tableau algorithms can no longer decide a TBox consistency test by constructing a tree-like model. Second, the nominals semantics impose global restrictions which means that arithmetic solutions used to satisfy a local restriction as in [3] must not invalidate the nominals semantics during the construction of a completion graph.

4 Algebraic Tableau Reasoning with Nominals

In [3] the hybrid algorithm uses the atomic decomposition technique to reduce reasoning about QCRs to linear inequation solving. With nominals the technique needs to handle the interaction between local and global numerical restrictions while preserving the nominals semantics. We first illustrate the preprocessing steps needed to allow the applicability of the atomic decomposition technique and show how we use this technique to reduce satisfying the nominals and QCRs semantics into linear inequation solving.

4.1 Preprocessing

For R, S in N_{R} and $C_{\mathcal{T}}, D$ $\mathcal{ALCOQ}$ concepts, we define a new concept operator $\forall(R\backslash S).D$, and a role implication operator, $R \sqsubseteq S$, needed to rewrite $C_{\mathcal{T}}$ before applying the calculus. These operators are based on set semantics such that given an interpretation $\mathcal{I}$, then $(\forall\ (R\backslash S).D)^{\mathcal{I}} = \{s \in \Delta^{\mathcal{I}} \mid \langle s,t\rangle \in R^{\mathcal{I}} \wedge \langle s,t\rangle \notin S^{\mathcal{I}} \Rightarrow t \in D^{\mathcal{I}}\}$ is satisfied, and $(R^{\mathcal{I}} \subseteq S^{\mathcal{I}})$ is satisfied for each role implication $R \sqsubseteq S \in \mathcal{R}$, with $\mathcal{R}$ a set of role implications and can be seen as a weak form of a role hierarchy $\mathcal{H}$ [10].

Algorithm 1 Given $A \in N_{\mathrm{C}}$, C , D $\mathcal{ALCOQ}$ concepts, $R \in N_{\mathrm{R}}$ and $\mathcal{R}$ the set of role implications, the following rewriting holds:

$$
\begin{array}{ll}
rw(A, N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow A \\
rw(\neg A, N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow \neg A \\
rw((C \sqcap D), N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow (rw(C, N_{\mathrm{R}}, \mathcal{R}) \sqcap rw(D, N_{\mathrm{R}}, \mathcal{R})) \\
rw((C \sqcup D), N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow (rw(C, N_{\mathrm{R}}, \mathcal{R}) \sqcup rw(D, N_{\mathrm{R}}, \mathcal{R})) \\
rw((\neg C), N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow rw(\dot{\neg} C, N_{\mathrm{R}}, \mathcal{R}) \\
rw(\forall R.C, N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow \forall R.rw(C, N_{\mathrm{R}}, \mathcal{R}) \\
rw((\geq nR.C), N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow (\geq nR' \sqcap \forall R'.rw(C, N_{\mathrm{R}} \cup \{R'\}, \mathcal{R} \cup \{R' \sqsubseteq R\})) \\
rw((\leq nR.C), N_{\mathrm{R}}, \mathcal{R}) & \longrightarrow (\leq nR' \sqcap \forall R'.rw(C, N_{\mathrm{R}} \cup \{R'\}, \mathcal{R}) \sqcap \\
 & \quad \forall\ (R\backslash R').rw(\dot{\neg} C, N_{\mathrm{R}} \cup \{R'\}, \mathcal{R} \cup \{R' \sqsubseteq R\}))
\end{array}
$$

Given $C_{\mathcal{T}}$, a set N_{R} of role names, and an empty set $\mathcal{R}$ of role implications, we re-write $C_{\mathcal{T}}$ using Algorithm 1 and Lemma 1 holds (see [4] for proofs and examples).

Lemma 1 (Preserving Satisfiability). *Rewriting concept expressions according to Algorithm 1 preserves satisfiability. Satisfying $C_{\mathcal{T}}$ consists of satisfying $rw(C_{\mathcal{T}}, N_{\mathrm{R}}, \mathcal{R})$ w.r.t. $\mathcal{R}$.*

Please note that each rewriting step for a number restriction introduces a fresh role name R' new in $\mathcal{T}$ and $\mathcal{R}$. Thus, N_{R} and $\mathcal{R}$ are extended. For example, rewriting $C_{\mathcal{T}}$ in (5) gives $C_{\mathcal{T}} = \begin{pmatrix} \neg o \sqcup (\geq 1S_{11} \sqcap \forall S_{11}.(\geq 1R_1 \sqcap \forall R_1.B \sqcap \forall R.C) \\ \sqcap \geq 1S_{21} \sqcap \forall S_{21}.(\geq 1R_2 \sqcap \forall R_2.B \sqcap \forall R.\neg C)) \end{pmatrix}$ with N_R, $\mathcal{R}$ extended such that $N_{\mathrm{R}} = \{R, R_1, R_2, S_1, S_{11}, S_2, S_{21}\}, \mathcal{R} = \{R_1 \sqsubseteq R,\ R_2 \sqsubseteq R,\ S_{11} \sqsubseteq S_1, S_{21} \sqsubseteq S_2\}$.

4.2 Arithmetic Reasoning with Nominals and QCRs

The atomic decomposition technique [14] has been used in [3, 8] to divide the sets of role fillers local to a domain element into mutually disjoint atomic sets and these set cardinalities are used as a bridging function to encode local numerical restrictions as inequations. With nominals, we extend the atomic decomposition technique to consider a global decomposition of the whole domain into disjoint sets such that each domain element belongs to exactly one set. Thus, one has to consider all possible interactions between local and global restrictions which can also be encoded into inequations. We illustrate how this works in the following.

The Atomic Decomposition For each role $R \in N_{\mathrm{R}}$ that is used in a number restriction or a QCR, a fresh sub-role R' introduced by the preprocessing enables a simple role hierarchy; R' can have only one super role R. Let $H(R)$ denote the set of role names for all sub-roles of R: $H(R) = \{R' \mid (R' \sqsubseteq R) \in \mathcal{R}\}$. We do not need to add R to $H(R)$ since it is always implied and does not occur in number restrictions anymore after preprocessing. For every role $R' \in H(R)$, the set of R'*-fillers* forms a subset of the set of R*-fillers* ($FIL(R') \subseteq FIL(R)$). We define $\overline{R'}$ to be the complement of R' w.r.t. $H(R)$, the set of $\overline{R'}$*-fillers* is then defined as $\overline{R'}$*-fillers* $=(FIL(R) \setminus FIL(R'))$.

Each subset P of $H(R)$ ($P \subseteq H(R)$) defines a unique set of role names that admits an interpretation $P^{\mathcal{I}}$ corresponding to the unique intersection of role fillers for the role names in P: $P^{\mathcal{I}} = \bigcap_{\mathrm{R}' \in P} FIL(R') \cap \bigcap_{\mathrm{R}'' \in (H(R) \setminus P)} FIL(\overline{R''})$. $P^{\mathcal{I}}$ cannot overlap with role fillers for role names that do not appear in P since it is assumed to overlap with their complement. This makes all $P^{\mathcal{I}}$ disjoint. For example, if $P_1 = \{R_1, R_2\}$, $P_2 = \{R_2, R_3\}$ and $H(R) = \{R_1, R_2, R_3\}$ this means that P_1 is the partition name for $FIL(R_1) \cap FIL(R_2) \cap FIL(\overline{R_3})$ which is equal to $P_1^{\mathcal{I}}$ and P_2 is the partition name for $FIL(R_2) \cap FIL(R_3) \cap FIL(\overline{R_1})$, and therefore, although $P_1 \cap P_2 = \{R_2\}$ we have $P_1^{\mathcal{I}} \cap P_2^{\mathcal{I}} = \emptyset$ (see [14]). Since $\mathcal{ALCOQ}$ does not allow concept expressions using role complements, no role complement will be explicitly used. For ease of presentation, we do not list the role complements in a partition name.

Each nominal $o \in N_{\mathrm{o}}$, $o^{\mathcal{I}}$ can interact with R*-fillers* for some R in N_{R} and become a subset of the set of R*-fillers* ($o^{\mathcal{I}} \subseteq FIL(R)$ if having for example $\geq 1R.C \sqcap \forall R.o$) . In order to handle such interactions we define the decomposition set $\mathcal{DS}$ of all possible role names and nominals as: $\mathcal{DS} = \bigcup_{R \in N_{\mathrm{R}}} H(R) \cup N_{\mathrm{o}}$. Let $\mathcal{P}$ be the set of partition names defined for the decomposition of $\mathcal{DS}$: $\mathcal{P} = \{P \mid P \subseteq \mathcal{DS}\}$. For $P \subseteq \mathcal{DS}$, $P^{\mathcal{I}} = \bigcup_{R \in P} FIL(R) \cap \{d \in \Delta^{\mathcal{I}} \mid d \in o^{\mathcal{I}}, o \in P \cap N_o\} \cap \{d \in \Delta^{\mathcal{I}} \mid d \in \neg o^{\mathcal{I}}, o \in N_o \setminus P\}$. Then $\mathcal{P}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ because it includes all possible domain elements which correspond to a nominal and/or a role filler; $\mathcal{P}^{\mathcal{I}} = \bigcup_{P \subseteq \mathcal{DS}} P^{\mathcal{I}}$.

Mapping Cardinalities to Variables We assign a variable name v for each partition name P such that v can be mapped to a non-negative integer value n using $\sigma : \mathcal{V} \rightarrow \mathbb{N}$ such that $\sigma(v)$ denotes the cardinality of $P^{\mathcal{I}}$. Let $\mathcal{V}$ be the set of all variable names and α: $\mathcal{V} \rightarrow \mathcal{P}$ be a one-to-one mapping between each partition name $P \in \mathcal{P}$ and a variable $v \in \mathcal{V}$ such that $\alpha(v) = P$, and if a non-negative integer n is assigned to v using σ then $\sigma(v) = n = \#P^{\mathcal{I}}$. Let V_S denote the set of variable names mapped to partitions for a role ($S \in N_R$) or a nominal ($S \in N_o$) then V_S is defined as $V_S = \{v \in \mathcal{V} \mid S \in \alpha(v)\}$.

Using Inequation Solving Given a partitioning $\mathcal{P}$ for all roles in N_R and nominals in N_o and a mapping α of variables, we can reduce a conjunction of $(\geq nR)$ and $(\leq mR)$ to a set of inequations and reason about the numerical restrictions using an inequation solver based on the following principles.

P1: Encoding Number Restrictions and Nominals Into Inequations. Since the partitions in $\mathcal{P}$ are mutually disjoint and the cardinality function is additive, a lower (upper) bound n (m) on the cardinality of the set of role fillers $FIL(S)$ for some role $S \in H(R)$ can be reduced to an inequation of the form $\sum_{v \in V_S} \sigma(v) \geq n$ ($\sum_{v \in V_S} \sigma(v) \leq m$). Thus, we can easily convert an expression of the form $(\geq nS)$ or $(\leq mS)$ into an inequation using ξ such that $\xi(S, \geq, n) = \sum_{v \in V_S} \sigma(v) \geq n$, and $\xi(S, \leq, m) = \sum_{v \in V_S} \sigma(v) \leq m$. The cardinality of a partition with a nominal can only be 1 based on the nominals semantics which is encoded into inequations using $\xi(o, \geq, 1)$ and $\xi(o, \leq, 1)$ for each nominal $o \in N_o$. In this way we make sure that the nominal semantics is preserved; there is one individual for each $o \in N_o$: $\#o^{\mathcal{I}} = 1$.

P2: Getting a Solution. Given a set ξ_a of inequations, an integer solution defines the mapping σ for each variable v occurring in ξ_a to a non-negative integer n denoting the cardinality of the corresponding partition. For example, assuming $\sigma(v) = 4$ and $\alpha(v) = \{R', R''\}$, this means that the corresponding partition $(\alpha(v))^{\mathcal{I}}$ must have 4 fillers; $\#(FIL(R') \cap FIL(R'')) = 4$. Additionally, by setting the objective function to minimize the sum of all variables a minimum number of role fillers is ensured at each level. σ then defines a distribution of individuals that is consistent with the numerical restrictions encoded in ξ_a and the hierarchy expressed in $\mathcal{R}$.

4.3 The Tableau

In general the satisfiability of numerical restrictions does not decide TBox consistency since logical restrictions expressed using logical operators $(\sqcap, \sqcup, \neg, \sqsubseteq)$ and using $\forall$ operators need also be satisfied. Numerical and logical restrictions share expressions and their satisfiability cannot be tested independently to decide a TBox consistency. For this purpose we propose a hybrid algorithm; we define a tableau for $\mathcal{ALCOQ}$ TBox consistency and describe the algorithm in the next section. Our tableau is different from other tableaux for $\mathcal{ALCOQ}$ by the way it ensures the semantics of the QCRs operator and the new operator $(\forall(R \backslash S))$ introduced after preprocessing an unfoldable $\mathcal{ALCOQ}$ TBox. We also define *clos(C)* to be the smallest set of concepts such that: (a) $C \in$ *clos(C)*, (b) if $D \in$ *clos(C)* then $\doteq\neg D \in$ *clos(C)*, (c) if $(E \sqcap D)$ or $(E \sqcup D) \in$ *clos(C)* then $E, D \in$ *clos(C)*, (d) if $(\forall R.D)$ or $(\forall R \backslash S.D) \in$ *clos(C)* then $D \in$ *clos(C)*. The set of relevant sub-concepts of a TBox $\mathcal{T}$ is then defined as $clos(\mathcal{T}) = clos(rw(C_{\mathcal{T}}, N_R, \mathcal{R}))$.

Definition 3. [Tableau] Given an unfolded $\mathcal{ALCOQ}$ TBox $\mathcal{T}$ rewritten by applying Algorithm 1, we define a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ as an abstraction of a model for $\mathcal{T}$ with $\mathbf{S}$ a non-empty set of individuals, $\mathcal{L} : \mathbf{S} \rightarrow 2^{clos(\mathcal{T})}$ a mapping between each individual and a set of concepts, and $\mathcal{E}$: $N_R \rightarrow 2^{S \times S}$ a mapping between each role and a set of pairs of individuals in $\mathbf{S}$. For all $s, t \in \mathbf{S}$, $A \in N_C$, C,$D \in$ *clos(T)*, $o \in N_o$, R, $S \in N_R$, and given the definition $R^{\mathcal{T}}(s) = \{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}(R)\}$, properties 1 - 10 must always hold:

1. $C_{\mathcal{T}} \in \mathcal{L}(s)$

2. If $A \in \mathcal{L}$(s) then $\neg A \notin \mathcal{L}$(s).
3. If $C \sqcap D \in \mathcal{L}$(s) then $C \in \mathcal{L}(s)$ and $D \in \mathcal{L}(s)$.
4. If $C \sqcup D \in \mathcal{L}$(s) then $C \in \mathcal{L}(s)$ or $D \in \mathcal{L}(s)$.
5. If $\forall S.C \in \mathcal{L}$(s) and $\langle s, t\rangle \in \mathcal{E}(S)$ then $C \in \mathcal{L}(t)$.
6. If $\forall(R \backslash S).C \in \mathcal{L}$(s) and $\langle s, t\rangle \in \mathcal{E}(R)$, and $\langle s, t\rangle \notin \mathcal{E}(S)$ then $C \in \mathcal{L}(t)$.
7. If $(\geq nR) \in \mathcal{L}$(s) then $\#R^{\mathcal{T}}(s) \geq n$.
8. If $(\leq mR) \in \mathcal{L}$(s) then $\#R^{\mathcal{T}}(s) \leq m$.
9. If $\langle s, t\rangle \in \mathcal{E}(R)$ and $R \sqsubseteq S \in \mathcal{R}$, then $\langle s, t\rangle \in \mathcal{E}(S)$.
10. For each $o \in N_{\mathrm{o}}$, $\#\{s \in \mathbf{S} \mid o \in \mathcal{L}(s)\} = 1$

Lemma 2. *An $\mathcal{ALCOQ}$ TBox $\mathcal{T}$ is consistent iff there exists a tableau for $\mathcal{T}$.*

Proof. The proof is similar to the one found in [10]. Property 6 of this tableau ensures that the semantics of the $\forall(R\backslash S).C$ operator is preserved. Property 9 ensures that the form of role hierarchy introduced at preprocessing is preserved, together with Properties 8 and 7 this property ensures that the semantics of the original (before rewriting) QCRs is preserved. Property 10 ensures that the semantics of nominals is preserved.

5 A Hybrid Algorithm for $\mathcal{ALCOQ}$

In this section, we describe a hybrid algorithm which decides the existence of a tableau for an unfolded $\mathcal{ALCOQ}$ TBox $\mathcal{T}$. Our algorithm is hybrid because it relies on tableau expansion rules working together with an inequation solver, the corresponding model is represented using a *compressed completion graph* (CCG). The CCG is different from the "so-called" completion graphs used in standard tableau algorithms for $\mathcal{ALCOQ}$ [9] in four ways. (I) it makes no distinction between a nominal node and a blockable node. (II) it allows the re-use of existing nodes instead of creating new similar ones. (III) it implements no blocking or merging of existing nodes. Finally, it uses a new labeling for nodes to collect and encode the number restrictions into inequations. In the context of this paper we use "completion graph" to refer to a CCG.

Definition 4. [Compressed Completion Graph] A *compressed completion graph* is a directed graph $G = (V, E, \mathcal{L}, \mathcal{L}_{\mathrm{E}})$. Each node $x \in V$ is labeled with two labels: $\mathcal{L}(x)$ and $\mathcal{L}_E(x)$, and each edge $\langle x, y\rangle \in E$ is labeled with a set, $\mathcal{L}(\langle x, y\rangle) \subseteq N_{\mathrm{R}}$, of role names.

$\mathcal{L}(x)$ denotes a set of concept expressions and role names such that $\mathcal{L}(x) \subseteq clos(\mathcal{T}) \cup \mathcal{P}$. By doing this, we not only label nodes based on the concept descriptions that they satisfy but also based on the partition they belong to. A partition name might include a nominal or a role name. We do not need to distinguish whether a nominal $o \in \mathcal{L}(x)$ is part of a partition name or a concept expression; in both cases x satisfies the nominal o. When a role name R appears in $\mathcal{L}(x)$ this means that x belongs to the partition for *R-fillers* and can therefore be used as an *R-filler*. This tagging is needed for the re-use of individual nodes.

$\mathcal{L}_{\mathrm{E}}(x)$ denotes a set ξ_x of inequations that must have a non-negative integer solution. The set ξ_x is the encoding of $(\geq nR)$ and $(\leq mR) \in \mathcal{L}(x)$. In order to make sure that numerical restrictions local for a node x are satisfied while the global restrictions carried with nominals are not violated, $\mathcal{L}_{\mathrm{E}}(x)$ is propagated from each node to all its successors. This makes sure that nominals are globally preserved while still satisfying the numerical restrictions at each level.

Given $\mathcal{T}$, we unfold $\mathcal{T}$ as outlined in Section 2 and build $C'_{\mathcal{T}}$ which is rewritten into $C_{\mathcal{T}} = rw(C'_{\mathcal{T}}, N_{\mathrm{R}}, \mathcal{R})$ and $\mathcal{P}$ is the corresponding atomic decomposition. To decide the consistency of $\mathcal{T}$ we need to test the consistency of $C_{\mathcal{T}}$ using $i \in N_{\mathrm{o}}$ new in $\mathcal{T}$ such that $i^{\mathcal{I}} \in {C_{\mathcal{T}}}^{\mathcal{I}}$ and every new individual satisfies $C_{\mathcal{T}}$.

The algorithm starts with the completion graph $G = (\{r_0\}, \emptyset\ , \mathcal{L}, \mathcal{L}_{\mathrm{E}})$. With $\mathcal{L}_{\mathrm{E}}(r_o) = \bigcup_{o \in N_{\mathrm{o}}} \{\xi(o, \leq, 1), \xi(o, \geq, 1)\}$ which is an encoding of the nominal semantics. The node r_0 is artificial and is not considered as part of the model, it is only used to process the numerical restrictions on nominals using the inequation solver which returns a distribution for them. The distribution of nominals (solution) is processed by the *fil*-Rule (see Fig. 1) which non-deterministically initializes the individual nodes for nominals. After at least one nominal is created, G is expanded by applying the expansion rules given in Fig. 1 until no more rules are applicable or when a clash occurs. No clash triggers or rules other then the *fil*-Rule apply to r_o.

Definition 5. [Clash] A node x in $(V \setminus \{r_0\})$ is said to contain a *clash* if:

- (i) $\{C, \neg C\} \subseteq \mathcal{L}(x)$, or
- (ii) a subset of inequations $\xi_x \subseteq \mathcal{L}_{\mathrm{E}}(x)$ does not admit a non-negative integer solution, this case is decided by the inequation solver, or
- (iii) for some $o \in N_{\mathrm{o}}$, $\#\{x \in (V \setminus \{r_0\}) | \, o \in \mathcal{L}(x)\} > 1$.

Definition 6. [Proxy node] A proxy node is a representative for the elements of each partition. Proxy nodes can be used since partitions are disjoint and all elements in a partition share similar restrictions (see Lemma 4 in [4] for a proof).

When no rules are applicable or there is a clash, a *completion graph* is said to be *complete*. When G is complete and there is no clash, this means that the numerical as well as the logical restrictions are satisfied (${C_{\mathcal{T}}}^{\mathcal{I}} \neq \emptyset$) and there exists a model for $\mathcal{T}$: the algorithm returns that $\mathcal{T}$ is consistent and inconsistent otherwise.

5.1 Strategy of Rule Application

Given a node x in the completion graph, the expansion rules in Figure 1 are triggered when applicable based on the following priorities:

- **Priority 1:** $\sqcap$-*Rule*, $\sqcup$-*Rule*, $\forall$-*Rule*, *ch-Rule*, $\leq$-*Rule*, $\geq$-*Rule*, *e-Rule*.
- **Priority 2:** *fil-Rule*.
- **Priority 3:** $\forall_{\setminus}$-*Rule*.

The rules with Priority 1 can be fired in arbitrary order. The *fil-Rule* has Priority 2 to ensure that all at-least and at-most restrictions for a proxy node x are encoded and satisfied by the inequation solver before creating any new proxy nodes. This justifies why role fillers or nominals are never merged nor removed from the graph; a distribution of role fillers and nominals either survives into a complete model or fails due to a clash. Also, assigning the *fil-Rule* Priority 2 helps in early clash detection in the case when the inequation solver detects a numerical clash even before new nodes are created. The $\forall_{\setminus}$-*Rule* has Priority 3. By giving this priority we ensure that the semantics of the $\forall(R \setminus S)$ operator are not violated. We allow the creation of all possible edges between a node and its successors before applying the $\forall(R \setminus S)$ operator semantics. This rule priority is needed to ensure the completeness of the algorithm (See Section 5.3 for proofs).

$\sqcap$-Rule	**If** $C \sqcap D \in \mathcal{L}(x)$, and $\{C, D\} \not\subseteq \mathcal{L}(x)$ **then** set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C, D\}$.
$\sqcup$-Rule	**If** $C \sqcup D \in \mathcal{L}(x)$, and $\{C, D\} \cap \mathcal{L}(x) = \emptyset$ **then** set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{E\}$ with $E \in \{C, D\}$.
$\forall$-Rule	**If** $\forall R.C \in \mathcal{L}(x)$ and there exists y such that $\mathcal{L}(\langle x, y\rangle) \cap (H(R) \cup \{R\}) \neq \emptyset$, with $C \notin \mathcal{L}(y)$ **then** set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$.
$\forall_\setminus$-Rule	**If** $\forall(R\setminus S).C \in \mathcal{L}(x)$, and there exists y such that $\mathcal{L}(\langle x, y\rangle) \cap (H(R) \cup \{R\}) \neq \emptyset$, and $\mathcal{L}(\langle x, y\rangle) \cap (H(S) \cup \{S\}) = \emptyset$ with $C \notin \mathcal{L}(y)$ **then** set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$.
$\leq$-Rule	**If** $(\leq nR) \in \mathcal{L}(x)$ and $\xi(R, \leq, n) \notin \mathcal{L}_{\mathrm{E}}(x)$ **then** set $\mathcal{L}_{\mathrm{E}}(x) = \mathcal{L}_{\mathrm{E}}(x) \cup \{\xi(R, \leq, n)\}$.
$\geq$-Rule	**If** $(\geq nR) \in \mathcal{L}(x)$ and $\xi(R, \geq, n) \notin \mathcal{L}_{\mathrm{E}}(x)$ **then** set $\mathcal{L}_{\mathrm{E}}(x) = \mathcal{L}_{\mathrm{E}}(x) \cup \{\xi(R, \geq, n)\}$.
ch-Rule	**If** there exists v occurring in $\mathcal{L}_{\mathrm{E}}(x)$ with $\{v \geq 1, v \leq 0\} \cap \mathcal{L}_{\mathrm{E}}(x) = \emptyset$ **then** set $\mathcal{L}_{\mathrm{E}}(x) = \mathcal{L}_{\mathrm{E}}(x) \cup \{V\}$, $V \in \{v \geq 1, v \leq 0\}$.
fil-Rule	**If** there exists v occurring in $\mathcal{L}_{\mathrm{E}}(x)$ with $\sigma(v) = m$ and $m > 0$, and there exists no y with $\alpha(v) \subseteq \mathcal{L}(y)$ **then** 1. create a new proxy node y, 2. set $\mathcal{L}(y) = \alpha(v) \cup \{C_{\mathcal{T}}\}$, 3. set $\mathcal{L}_{\mathrm{E}}(y) = \mathcal{L}_{\mathrm{E}}(x)$.
e-Rule	**If** $(\bowtie nR) \in \mathcal{L}(x)$, $n \geq 1$, and there exists y with $R \in \mathcal{L}(y)$ and $R \notin \mathcal{L}(\langle x, y\rangle)$ **then** set $\mathcal{L}(\langle x, y\rangle) = \mathcal{L}(\langle x, y\rangle) \cup \{R\}$, and if $\mathcal{L}_{\mathrm{E}}(x) \not\subseteq \mathcal{L}_{\mathrm{E}}(y)$ then set $\mathcal{L}_{\mathrm{E}}(y) = \mathcal{L}_{\mathrm{E}}(y) \cup \mathcal{L}_{\mathrm{E}}(x)$.

Fig. 1. Expansion rules for $\mathcal{ALCOQ}$

5.2 Explaining the Rules

The $\sqcap$-Rule, $\sqcup$-Rule and the $\forall$-Rule rules are similar to the ones in the standard tableau rules for $\mathcal{ALC}$ [2].

$\forall_\setminus$-Rule. This rule is used to ensure the semantics of the new operator $\forall(R\setminus S).D$ (defined in preprocessing) by making sure that all *R-fillers* are labelled, and together with the *ch*-Rule (see explanation below) it has the same effect as the *choose*-rule in [2] needed to detect the unsatisfiability of concepts like $((\geq 3R.C) \sqcap (\leq 1R.D) \sqcap (\leq 1R.\neg D))$ (See Example 2 in [4] for details).

$\leq$-Rule and **$\geq$-Rule.** These rules encode the numerical restrictions in the label $\mathcal{L}$ of a node x into a set (ξ_x) of inequations maintained in $\mathcal{L}_{\mathrm{E}}(x)$ (P1 in Section 4.2). An inequation solver is always active and is responsible for finding a non-negative integer solution σ for ξ_x (P2 in Section 4.2) or triggering a clash if no solution is possible (see Def. 5). If the inequations added by these rules do not trigger a clash, then the encoded number restrictions can be satisfied by a possible distribution of role fillers.

***ch*-Rule.** This rule is used to check for empty partitions. Given a set of inequations in the label ($\mathcal{L}_{\mathrm{E}}$) of a node x and a variable v such that $\alpha(v) = P$ and $P \in \mathcal{P}$ we distinguish between two cases:

- (i) The case when $P^{\mathcal{I}}$ must be empty ($v \leq 0, \alpha(v) = P$); this can happen when restrictions of elements assigned to this partition trigger a clash. For instance, if

$\{\forall R_1.A, \forall R_2.\neg A\} \subseteq \mathcal{L}(x)$, $v \geq 1 \in \mathcal{L}_{\mathrm{E}}(x)$ and $P = \{R_1, R_2\}$ then a node y assigned to P with $\{R_1, R_2\} \subseteq \mathcal{L}(\langle x, y\rangle)$ triggers a clash $\{A, \neg A\} \subseteq \mathcal{L}(y)$ and $v \leq 0$ is enforced.

– (ii) The case when $P^{\mathcal{I}}$ must have at least one element ($1 \leq m \leq \sigma(v)$); if $P^{\mathcal{I}}$ can have at least one element without causing any logical clash, this means that we can have m elements also in $P^{\mathcal{I}}$ without a clash. Therefore, when creating nodes (using the *fil*-Rule) of corresponding partitions, it is sufficient to create one proxy node for each partition (see Lemma 4 in [4] for a proof).

Since the inequation solver is unaware of logical restrictions of filler domains we allow an explicit distinction between cases (i) and (ii). We do this by non-deterministically assigning ≤ 0 or ≥ 1 for each variable v occurring in $\mathcal{L}_{\mathrm{E}}(x)$.

***fil*-Rule**. This rule is used to generate individual nodes depending on the distribution (σ) returned by the inequation solver. The rule is fired for every non-empty partition P using $\sigma(v)$. It generates one proxy node as the representative for the m elements assigned to $P^{\mathcal{I}}$ by the inequation solver. The proxy individual is tagged with its partition name using $\alpha(v)$ in its label, $C_{\mathcal{T}}$ is also added to its label to make sure that every node created by the *fil*-Rule also satisfies $C_{\mathcal{T}}$.

***e*-Rule.** This rule connects a node x to a proxy individual y representing *R-fillers* of x by adding the edge for R between x and y. For instance, if $(\geq 2R) \in \mathcal{L}(x)$ and there exists a node y assigned to $P^{\mathcal{I}}$ such that $R \in P$ and $P \subseteq \mathcal{L}(y)$, this means that y can be used as an *R-filler* of x. Therefore, the *e*-Rule creates/updates the edge $\langle x, y\rangle$ with $R \in \mathcal{L}(\langle x, y\rangle)$. The node y can be re-used to satisfy another $(\geq nR$. For instance, if we have another node x_1 with $(\geq 2R) \in \mathcal{L}(x_1)$, y is re-used and R is added to the edge $\langle x_1, y\rangle$.

5.3 Proofs

The soundness, completeness and termination of the algorithm presented in this paper are consequences of Lemmas 2, 3, 4, 5, and Lemma 6.

Lemma 3. *Given an unfoldable TBox $\mathcal{T}$ and its complete and clash-free completion graph G. Let x be a node in G, $C, D \in N_{\mathrm{C}}$, $R \in N_{\mathrm{R}}$, we define Num(x) = $\{E \in \mathcal{L}(x) \mid E$ is of the form $\geq nR$, $\leq mR\}$ as the set of at-least and at-most restrictions to be satisfied for x. A solution σ for the encoding ξ_x of Num(x) is valid w.r.t $\mathcal{T}$: (i) it does not violate Num(y) for a node y in G, (ii) it does not violate a restriction implied by any operator used in $\mathcal{T}$, (iii) it does not violate the hierarchy $\mathcal{R}$ introduced during preprocessing.*

Lemma 4 (Termination). *When started with an unfoldable TBox $\mathcal{T}$ using the DL $\mathcal{ALCOQ}$, the proposed algorithm terminates.*

Lemma 5 (Soundness). *If the expansion rules can be applied to $\mathcal{T}$ such that they yield a complete and clash-free completion graph, then $\mathcal{T}$ has a tableau.*

Lemma 6 (Completeness). *If $\mathcal{T}$ has a tableau, then the expansion rules can be applied to $\mathcal{T}$ such that they yield a complete and clash-free completion graph.*

Proof. The proofs for Lemmas 3, 4, 5, and Lemma 6 can be found in [4].

6 Discussion

The work presented in this paper not only extends our work in [3] to handle nominals and unfoldable TBoxes. Moreover, it reduces the number of elements created and the number of expansion rules triggered, by using one proxy element as in [6] to represent a partition's elements. Proxy elements are tagged by a partition name to enable their re-use. A different form of individual re-use is used in [11] and is based on non-deterministic guessing. The calculus itself is not an optimization technique; a naïve implementation is by no means better then any other naïve implementation of standard DL reasoning algorithms proposed so far. However, by integrating arithmetic reasoning, the calculus is more informed and enjoys characteristics that make it more open to address sources of inefficient DL reasoning such as those related to the creation of large models and the interaction between nominals and QCRs.

The key intuition in our approach is to apply the atomic decomposition technique on the set of all nominals and all role fillers. By doing this, the decomposition reaches all domain elements and computes all possible groupings of these elements into mutually disjoint partitions. In a sense it performs a semantic split for groups of individuals and not necessarily for each individual (which is the case with standard tableau algorithms) using the *ch*-Rule rule.

Unfortunately the number of partitions is exponential to the size of the input (number of nominals and QCRs) which results in an exponential blow up of variables. Naïve applications of the ch-Rule give a double exponential worst case algorithm. However, many of these partitions will not be assigned any individuals. For instance, in most of the cases the partitions including more then one nominal will be empty assuming that nominals are disjoint. As a default, one can assume that all partitions are empty and use only those needed to satisfy the QCRs. In a sense, the inequation solver only allocates and deals with non-zero variables and the variables not considered are assumed to be zero. Which means that in the average case the algorithm does not have to deal with inequations containing an exponential number of variables (see [5] for more details).

It has been shown in [8, 5] that extending a DL reasoning algorithm with an arithmetic component can dramatically improve the average case performance in the case of the DL $\mathcal{SHQ}$. We conjecture that the calculus presented in this paper can enable similar performance improvements once equipped with adequate optimizations. In particular, it seems to be amenable to the optimizations used in [5] and the ones discussed in [14]. A prototype implementation for the calculus presented in this paper is part of ongoing work where we also extend the calculus to handle GCIs, transitive roles and role hierarchies. Empirical evaluation will be reported in a separate paper.

References

1. Baader, F., Buchheit, M., and Hollunder, B. Cardinality restrictions on concepts. *Artificial Intelligence 88*, 1-2 (1996), 195–213.
2. Baader, F., and Sattler, U. An overview of tableau algorithms for description logics. *Studia Logica 69* (2001), 5–40.
3. Faddoul, J., Farsinia, N., Haarslev, V., and Möller, R. A hybrid tableau algorithm for $\mathcal{ALCQ}_{\mathcal{I}}$. In *Proc. of the 2008 Int. Workshop on Description Logics, also in 18th European Conference on Artificial Intelligence (ECAI 2008)* (2008), pp. 725–726.

4. Faddoul, J., Haarslev, V., and Möller, R. Hybrid reasoning for description logics with nominals and qualified number restrictions. Tech. rep., Institute for Software Systems (STS), Hamburg University of Technology, 2008. http://www.sts.tu-harburg.de/tech-reports/papers.html.
5. Farsinia, N. Combining integer programming and tableau-based reasoning: A hybrid calculus for the description logic $\mathcal{SHQ}$. Master's thesis, Concordia University, 2008.
6. Haarslev, V., and Möller, R. Optimizing reasoning in description logics with qualified number restrictions. In *Description Logics* (2001).
7. Hollunder, B., and Baader, F. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conference on Principles of Knowledge Representation and Reasoning,KR-91* (Boston (USA), 1991), pp. 335–346.
8. Haarslev, V., Timmann, M., and Möller, R. Combining tableaux and algebraic methods for reasoning with qualified number restrictions. In *Description Logics* (2001), pp. 152–161.
9. Horrocks, I., and Sattler, U. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)* (2001), Morgan Kaufmann, Los Altos, pp. 199–204.
10. Horrocks, I., and Sattler, U. A tableaux decision procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning* (2007), 249–276.
11. Motik, B., and Horrocks, I. Individual reuse in description logic reasoning. In *IJCAR* (2008), pp. 242–258.
12. Motik, B., Patel-Schneider, P. F., and Cuenca Grau, B. OWL 2 web ontology language: Direct semantics. Latest version available at http://www.w3.org/TR/owl2-semantics/.
13. Motik, B., Shearer, R., and Horrocks, I. Optimized reasoning in description logics using hypertableaux. In *(CADE-21)* (2007), vol. 4603 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 67–83.
14. Ohlbach, H. J., and Koehler, J. Modal logics description logics and arithmetic reasoning. *Artificial Intelligence 109*, 1-2 (1999), 1–31.
15. Parsia, B., Cuenca Grau, B., and Sirin, E. From wine to water: Optimizing description logic reasoning for nominals. In *Proc. of the 10th Int. Conference on Principles of Knowledge Representation and Reasoning (KR)* (2006), pp. 90–99.
16. Tobies, S. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research 12* (2000), 199–217.
17. Wolstencroft, K., Brass, A., Horrocks, I., Lord, P., Sattler, U., Stevens, R., and Turi, D. A little semantic web goes a long way in biology. In *4th Int. Semantic Web Conference* (Ireland, 2005), vol. 3792, pp. 786–800.
18. Kazakov, Y., and Motik, B. *A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$*, vol. 4130/2006. Springer Berlin / Heidelberg, 2006, pp. 662–677.