

Algebraic Reasoning for \mathcal{SHIQ}

Laleh Roosta Pour and Volker Haarslev

Concordia University, Montreal, Quebec, Canada

Abstract. We present a hybrid tableau calculus for the description logic \mathcal{SHIQ} that decides ABox consistency and uses an algebraic approach for more informed reasoning about qualified number restrictions (QNRs). Benefiting from integer linear programming and several optimization techniques to deal with the interaction of QNRs and inverse roles, our approach provides a more informed calculus. A prototype reasoner based on the hybrid calculus has been implemented that decides concept satisfiability for \mathcal{ALCHIQ} . We provide a set of benchmarks that demonstrate the effectiveness of our hybrid reasoner.

1 Introduction

It is well known that standard tableau calculi for reasoning with qualified number restrictions (QNRs) in description logics (DLs) have no explicit knowledge about set cardinalities implied by QNRs. This lack of information causes significant performance degradations for DL reasoners if the numbers occurring in QNRs are increased. Over the last years a family of hybrid tableau calculi has been developed that address this inefficiency by integrating integer linear programming (ILP) with DL tableau methods, where ILP is used to reason about these set cardinalities. We developed hybrid calculi for the DLs \mathcal{ALCQ} [3], \mathcal{SHQ} [5], and \mathcal{SHOQ} [4, 2]. In this paper we present a new calculus that decides ABox consistency for the DL \mathcal{SHIQ} , which extends \mathcal{SHQ} with inverse roles. This new calculus is a substantial extension of the one for \mathcal{SHQ} [5] since the interaction between inverse roles and QNRs results in back propagation of information possibly adding new back-propagated QNRs, which, in turn, possibly require a conservative extension of solutions obtained by using ILP.

Informally speaking this line of research is based on several principles: (i) role successors are semantically partitioned into disjoint sets such that all members of one set are indistinguishable w.r.t. to their restrictions; (ii) cardinalities of partitions are denoted by non-negative integer variables and the cardinality of a union of partitions can be expressed by the sum of the corresponding partition variables; (iii) all members of a non-empty partition are represented by a proxy element [6] associated with the corresponding partition variable; (iv) cardinality restrictions on a set of role successors imposed by QNRs are encoded as a set of linear inequations; (v) the satisfiability of a set of QNRs is mapped to the problem whether the corresponding system of linear inequations has a non-negative integer solution and the involved proxy elements do not lead to a logical contradiction. This approach is better informed than traditional tableau methods because (i) (implied) set cardinalities are represented using ILP that is independent of the values occurring in QNRs; (ii) the tableau part of the hybrid calculus becomes more efficient because a set of indistinguishable role successors is represented by one proxy; (iii) the partitioning of role successors supports semantic branching on partition cardinalities and more refined dependency-directed backtracking.

2 Preliminaries

In this section we briefly describe syntax and semantics of \mathcal{SHIQ} and its basic inferences services. Furthermore, we define a rewriting that reduces \mathcal{SHIQ} to \mathcal{SHIN}^\wedge in order to apply the atomic decomposition technique [10], which is a basic foundation of our calculus. Let N_C be a set of concept names, N_R a set of role names, $N_{TR} \subseteq N_R$ a set of transitive role names, $N_{SR} \subseteq N_R$ a set of non-transitive role names with $N_{SR} \cap N_{TR} = \emptyset$. The set of roles is defined as $N_{RS} = N_R \cup \{R^- \mid R \in N_R\}$. We define a function Inv such that $Inv(R) = R^-$ if $R \in N_R$, and $Inv(R) = S$ if $R = S^-$. For a set of roles $RO = \{R_1, \dots, R_n\}$, $Inv(RO) = \{Inv(R_1), \dots, Inv(R_n)\}$. A *role hierarchy* \mathcal{R} is a set of axioms of the form $R \sqsubseteq S$ where $R, S \in N_{RS}$ and \sqsubseteq_* is transitive-reflexive closure of \sqsubseteq over $\mathcal{R} \cup \{Inv(R) \sqsubseteq Inv(S) \mid R \sqsubseteq S \in \mathcal{R}\}$. R is called a sub-role of S and S a super-role of R if $R \sqsubseteq_* S$. A role $R \in N_{SR}$ is called simple if R is neither transitive nor has a transitive sub-role. The set of \mathcal{SHIQ} concepts is the smallest set such that (i) every concept name is a concept, and (ii) if A is a concept name, C and D are concepts, R is a role, S is a simple role, $n, m \in \mathbb{N}, n \geq 1$, then $C \sqcap D, C \sqcup D, \neg C, \forall R.C, \exists R.C, \geq nS.C$, and $\leq mS.C$ are also concepts. We consider \top (\perp) as abbreviations for $A \sqcup \neg A$ ($A \sqcap \neg A$). A general concept inclusion axiom (GCI) is an expression of the form $C \sqsubseteq D$ with C, D concepts. A \mathcal{SHIQ} TBox \mathcal{T} w.r.t. a role hierarchy \mathcal{R} is a set of GCIs.

Let I be a set of individual names. A \mathcal{SHIQ} ABox \mathcal{A} w.r.t. a role hierarchy \mathcal{R} is a finite set of assertions of the form of $a : C, \langle a, b \rangle : R$, and $a \neq b$ with $I_A \subseteq I$ the set of individuals occurring in \mathcal{A} and $a, b \in I_A$ and R a role. We assume an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the non-empty set $\Delta^{\mathcal{I}}$ is the domain of \mathcal{I} and $\cdot^{\mathcal{I}}$ is an interpretation function which maps each concept to a subset of $\Delta^{\mathcal{I}}$ and each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Semantics and syntax of the DL \mathcal{SHIQ} are presented in [8].

An interpretation \mathcal{I} holds for a role hierarchy \mathcal{R} iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$. An interpretation \mathcal{I} satisfies a TBox \mathcal{T} iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every GCI $C \sqsubseteq D \in \mathcal{T}$. An interpretation \mathcal{I} satisfies an ABox \mathcal{A} if it satisfies \mathcal{T} and \mathcal{R} and all assertions in \mathcal{A} such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ if $a : C \in \mathcal{A}$, $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ if $\langle a, b \rangle : R \in \mathcal{A}$, and $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b \in \mathcal{A}$. Such an interpretation is called a model of \mathcal{A} . An ABox \mathcal{A} is consistent iff there exists a model \mathcal{I} of \mathcal{A} . A concept description C is satisfiable iff $C^{\mathcal{I}} \neq \emptyset$.

Let E be a concept expression, then $clos(E)$ defines the usual closure of all concept names occurring in E . Therefore, for a TBox \mathcal{T} if $C \sqsubseteq D \in \mathcal{T}$, then $clos(C) \subseteq clos(D)$ and $clos(D) \subseteq clos(\mathcal{T})$. Likewise for an ABox \mathcal{A} , if $(a : C) \in \mathcal{A}$ then $clos(C) \subseteq clos(\mathcal{A})$. Inspired by [10] we use a satisfiability-preserving rewriting to replace QNRs with unqualified ones. This rewriting uses a new role-set difference operator $\forall(R \setminus R').C$ for which $(\forall(R \setminus R').C)^{\mathcal{I}} = \{x \mid \forall y : \langle x, y \rangle \in R^{\mathcal{I}} \setminus R'^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$. We name the new language \mathcal{SHIN}^\wedge . We have $(\geq nR)^{\mathcal{I}} = (\geq nR.\top)^{\mathcal{I}}$ and $(\leq nR)^{\mathcal{I}} = (\leq nR.\top)^{\mathcal{I}}$. Considering $\neg C$ as the standard negation normal form (NNF) of $\neg C$ we define a recursive function unQ which rewrites \mathcal{SHIQ} concept descriptions and assertions into \mathcal{SHIN}^\wedge .

Definition 1 (unQ). Let R' be a new role in N_R with $\mathcal{R} := \mathcal{R} \cup \{R' \sqsubseteq R\}$ for each transformation. unQ rewrites the axioms as follows: $unQ(C) := C$ if $C \in N_C$, $unQ(\neg C) := \neg C$ if $C \in N_C$ otherwise $unQ(\neg C)$, $unQ(\forall R.C) := \forall R.unQ(C)$, $unQ(C \sqcap D) := unQ(C) \sqcap unQ(D)$, $unQ(C \sqcup D) := unQ(C) \sqcup unQ(D)$,

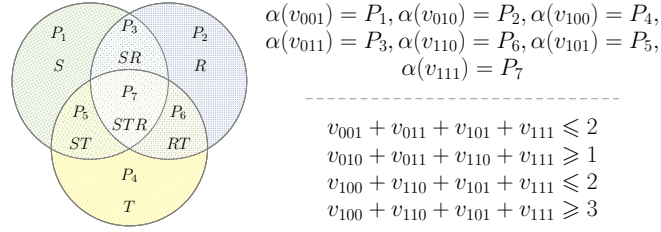


Fig. 1. Atomic decomposition for $\leq 2S \sqcap \geq 1R \sqcap \leq 2T \sqcap \geq 3T$

$unQ(\geq nR.C) := \geq nR' \sqcap \forall R'.unQ(C)$, $unQ(\leq nR.C) := \leq nR' \sqcap \forall(R \setminus R')$.
 $unQ(\dot{\neg}C)$, $unQ(a : C) := a : unQ(C)$, $unQ(\langle a, b \rangle : R) := \langle a, b \rangle : R$,
 $unQ(a \neq b) := a \neq b$.

Note that this rewriting generates a unique new role for each QNR. For instance, $unQ(\geq nR.C \sqcap \leq mR.C \sqcap \geq kR.D)$ is rewritten to $\geq nR_1 \sqcap \forall R_1.C \sqcap \leq mR_2 \sqcap \forall(R \setminus R_2). \neg C \sqcap \geq kR_3 \sqcap \forall R_3.D$ and $\{R_1 \sqsubseteq R, R_2 \sqsubseteq R, R_3 \sqsubseteq R\} \subseteq \mathcal{R}$. \mathcal{SHIQ}^{\wedge} is not closed under negation due to the fact that $unQ(\leq nR.C)$ itself creates a negation which must be in NNF before further applying unQ . In order to avoid the whole negating problem for the concept description generated by $unQ(\leq nR.C)$ and $unQ(\geq nR.C)$ the calculus makes sure that the application of unQ starts from the innermost part of an axiom, therefore such concept descriptions will never be negated.

Atomic decomposition A so-called atomic decomposition for reasoning about sets was proposed in [10] and later applied to DLs for reasoning about sets of role fillers (see Def. 3 for role fillers). For instance, for the concept description $\leq 2S \sqcap \geq 1R \sqcap \leq 2T \sqcap \geq 3T$ we get 7 disjoint partitions shown in the left part of Fig. 1, where partition P_1 represents S -fillers that are neither R nor T fillers and P_5 represents fillers in the intersection of S and T that are not R -fillers, etc. For example, due to the disjointness of $P_i, 1 \leq i \leq 7$, the cardinality of all S -fillers can be expressed as $|P_1| + |P_3| + |P_5| + |P_7|$ ($|\cdot|$ denotes the cardinality of a set). The satisfiability of the above-mentioned concept descriptions can now be defined by finding a non-negative integer solution for the inequations shown at the bottom-right part of Fig. 1, where v_i denotes the cardinality of P_i with i in unary encoding.

3 SHIQ ABox Calculus

In this section we introduce our calculus and the tableau completion rules for \mathcal{SHIQ} .

Definition 2 (Completion forest). *The algorithm generates a model consisting of a set of arbitrarily connected individuals in $I_{\mathcal{A}}$ as the roots of completion trees. Ignoring the connections between roots, the created model is a forest $\mathcal{F} = (V, E, \mathcal{L}, \mathcal{L}_E, \mathcal{L}_I)$ for a \mathcal{SHIQ} ABox \mathcal{A} . Every node $x \in V$ is labeled by $\mathcal{L}(x) \subseteq \text{clos}(\mathcal{A})$ and $\mathcal{L}_E(x)$ as a set of inequations of the form $\sum_{i \in \mathbb{N}} v_i \bowtie n$ with $n \in \mathbb{N}$ and $\bowtie \in \{\geq, \leq\}$ and variables $v_i \in \mathcal{V}$. Each edge $\langle x, y \rangle \in E$ is labeled by the set $\mathcal{L}(\langle x, y \rangle) \subseteq N_R$. For each node x , $\mathcal{L}_I(x)$ is defined to keep an implied back edge for x equivalent to $\text{Inv}(\mathcal{L}(\langle y, x \rangle))$, where y is a parent of x (see Def. 4). For the nodes with no parents (root nodes) \mathcal{L}_I will be the empty set.*

Definition 3 (-successor, -predecessor, -neighbour, -filler). *Given a completion tree, for nodes x and y with $R \in \mathcal{L}(\langle x, y \rangle)$ and $R \sqsubseteq_* S$, y is called S -successor of x and x is $\text{Inv}(S)$ -predecessor of y . If y is an S -successor or an $\text{Inv}(S)$ -predecessor of x*

then y is called an S -neighbor of x . In addition, if $R \in \mathcal{L}(\langle x, y \rangle)$ then y is an R -filler (role-filler) for x . The R -fillers of x are defined as $Fil(x, R) = \{y \mid \langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \in R^{\mathcal{I}}\}$.

Definition 4 (Precedence). Due to the existence of inverse roles for each pair of individuals x, y , $R \in \mathcal{L}(\langle x, y \rangle)$ imposes $Inv(R) \in \mathcal{L}(\langle y, x \rangle)$. A global counter PR keeps the number of nodes and is increased by one when a new node x is created, setting $PR_x = PR$. Hence, each node is ranked with a PR . A successor of x with the lowest PR is called parent (parent successor) of x and others are called its children. Accordingly, a node x has a lower precedence than a node y if x has a lower rank compared to y . Also, each node has a unique rank and no two nodes have the same rank.

Definition 5 (ξ). Assuming a set of variables \mathcal{V} , a unique variable $v \in \mathcal{V}$ is associated with a set of role names RV_v . Let $\mathcal{V}^R = \{v \in \mathcal{V} \mid R \in RV_v\}$ be the set of all variables which are related to a role R . The function ξ maps number restrictions to inequations such that $\xi(R, \bowtie, n) := (\sum_{v_i \in \mathcal{V}^R} v_i) \bowtie n$.

Definition 6 (Distinct partitions). R_x is defined as the set of related roles for x such that $R_x = \{S \mid \{\xi(S, \geq, n), \xi(S, \leq, m)\} \cap \mathcal{L}_E(x) \neq \emptyset\}$. A partitioning \mathcal{P}_x is defined as $\mathcal{P}_x = \bigcup_{P \subseteq R_x} \{P\} \setminus \{\emptyset\}$. For a partition $P_x \in \mathcal{P}_x$, $P_x^{\mathcal{I}} = (\bigcap_{S \in P_x} Fil^{\mathcal{I}}(x, S)) \setminus (\bigcup_{S \in (R_x \setminus P_x)} Fil^{\mathcal{I}}(x, S))$ with $Fil^{\mathcal{I}}(x, S) = \{y^{\mathcal{I}} \mid y \in Fil(x, S)\}$. Let α be a mapping $\alpha : \mathcal{V} \leftrightarrow \mathcal{P}_x$ for a node x , each variable $v \in \mathcal{V}$ is assigned to a partition $P_x \in \mathcal{P}_x$ such that $\alpha(v) = P_x$.

The definition clearly demonstrates that the fillers of x related to the roles of partition P_x are not the fillers of the roles in $R_x \setminus P$ (other partitions). Therefore, by definition the fillers of x associated with the partitions in \mathcal{P}_x are mutually disjoint w.r.t. the interpretation \mathcal{I} . An arithmetic solution is defined using the function $\sigma : \mathcal{V} \rightarrow \mathbb{N}$ mapping each variable in \mathcal{V} to a non-negative integer. Let \mathcal{V}_x be the set of all variables assigned to a node x such that $\mathcal{V}_x = \{v_i \in \mathcal{V} \mid v_i \in \mathcal{L}_E(x)\}$, a solution Ω for a node x is $\Omega(x) := \{\sigma(v) = n \mid n \in \mathbb{N}, v \in \mathcal{V}_x\}$. The inequation solver uses an objective function to determine whether to minimize the solution or maximize it. We minimize the solution in order to keep the size of the forest small.

The example in Fig. 1 depicts the process of finding an arithmetic solution in more detail. Let $\mathcal{L}(x) = \{\leq 2S, \geq 1R, \leq 2T, \geq 3T\}$ be the label of node x . Applying the atomic decomposition for the related roles $R_x = \{S, R, T\}$ results in seven disjoint partitions such that $\mathcal{P}_x = \{P_i \mid 1 \leq i \leq 7\}$ where $P_1 = \{S\}$, $P_2 = \{R\}$, $P_4 = \{T\}$, $P_3 = \{S, R\}$, $P_5 = \{S, T\}$, $P_6 = \{R, T\}$, $P_7 = \{R, S, T\}$ as shown in Fig. 1. In order to simplify the mapping between variables and partitions, each bit in the binary coding of a variable index represents a specific role in R_x . Therefore, in this example the first bit from right represents S , the second R , and the last T . Since $|R_x| = 3$, the number of variables in \mathcal{V}_x becomes $2^3 - 1$. The mapping of variables and the resulting inequations in $\mathcal{L}_E(x)$ are also shown in Fig. 1.

Definition 7 (Node Cardinality). The cardinality associated with proxy nodes is defined by the mapping $card : \mathcal{V} \rightarrow \mathbb{N}$.

Since the hybrid algorithm requires to have all numerical restrictions encoded as a set of inequations, three functions are defined to map number restrictions (NRs) to inequations and/or further constrain variables. Function ξ (see Def. 5) is used in the \geq -Rule and \leq -Rule as shown in Fig. 2. Function ζ and ς also add new inequations to the label \mathcal{L}_E of a node and modify the variables. The functions ζ and ς are respectively used in the IBE -Rule and $reset_{IBE}$ -Rule as shown in Fig. 2.

Definition 8 (ζ). For a set of roles RO and $k \in \mathbb{N}$, the function $\zeta(RO, k)$ maps number restrictions to inequations via the function ξ for each $R_j \in RO$. $\zeta(RO, k)$ returns a set of inequations such that $\zeta(RO, k) = \{\xi(R_j, \geq, k) \mid R_j \in RO\} \cup \{\xi(R_j, \leq, k) \mid R_j \in RO\}$. For $v \in \mathcal{V}^{R_j}$, if $R_j \in \alpha(v) \wedge \alpha(v) \not\subseteq RO$ then $v \leq 0$ is also returned.

Definition 9 (ς). For a set of roles RO and $k \in \mathbb{N}$, the function $\varsigma(RO, k)$ maps number restrictions to inequations via the function ξ for each $R_j \in RO$. $\varsigma(RO, k)$ returns a set of inequations such that $\varsigma(RO, k) = \{\xi(R_j, \geq, k) \mid R_j \in RO\} \cup \{\xi(R_j, \leq, k) \mid R_j \in RO\}$. For $v \in \mathcal{V}^{R_j}$, if $RO = \alpha(v)$ then $v = k$ is also returned.

Definition 10 (Proper Sub-Role). For each role in existing number restrictions a set will be assigned which contains a specific type of sub-role called proper sub-role. A proper sub-role $\mathfrak{R}(R)$ for a role R is defined as $\mathfrak{R}(R) = \{R_i \mid (R \in N_R \cup Inv(R)) \wedge R_i \sqsubseteq R\}$.

This makes specializing the edges between nodes possible. Therefore, in our algorithm when a role set is assigned to $\mathcal{L}(\langle x, y \rangle)$ a new proper sub-role S_i will be created for each role $S \in \mathcal{L}(\langle x, y \rangle)$, where $\mathfrak{R}(S) = \mathfrak{R}(S) \cup \{S_i\}$, and will be assigned to the edge label. A role in $\mathfrak{R}(S)$ cannot have any proper sub-role. Only roles that occur in number restrictions and their inverses can have proper sub-roles. Since these proper sub-roles do not appear in the logical label of nodes, they do not violate the correctness of our algorithm.

Definition 11 (Blocked Node). Since *SHIQ* does not have the finite model property pair-wise blocking [7] is used. Node y is blocked by node x , also called witness, if $\mathcal{L}(x) = \mathcal{L}(y)$ and for their successors y', x' , $\mathcal{L}(y') = \mathcal{L}(x')$ and $\mathcal{L}(\langle x, x' \rangle) = \mathcal{L}(\langle y, y' \rangle)$. Also, unreachable nodes which were discarded from the forest (due to the application of the reset-Rule or reset_{IBE}-Rule) are called blocked. In order to detect blocked nodes, all roles in a proper sub-role of role R are considered equivalent to R .

Definition 12 (Clash Triggers). A node x contains a clash if $\{A, \neg A\} \subseteq \mathcal{L}(x)$ (logical clash) or $\mathcal{L}_E(x)$ does not have a non-negative integer solution (arithmetic clash).

The interaction between the tableau rules and the inequation solver is similar to the clash triggers. No particular rule is needed to invoke the inequation solver. For each $\mathcal{L}_E(x)$, there is always a solution (if there exists any) otherwise a clash occurs. If a variable changes or $\mathcal{L}_E(x)$ is extended, a new solution will be calculated automatically. The completion rules for a *SHIQ* ABox are shown in Fig. 2, listed in decreasing priority from top to bottom. Rules in the same cell have the same priority. Rules with lower priorities cannot be applied to a node x , which is not blocked, if there exists any rule with a higher priority still applicable to it. Among the completion rules in Fig. 2, the \sqcap -Rule, \sqcup -Rule, \forall -Rule, \forall_+ -Rule are the same as in standard tableaux. The merge-Rule, \forall_- -Rule, *ch*-Rule, \geq -Rule, \leq -Rule are similar to [5].

\geq -Rule and \leq -Rule: all number restrictions from $\mathcal{L}(x)$ are collected via these two rules. The function ξ maps them to inequations according to the proper atomic decomposition and adds them to $\mathcal{L}_E(x)$.

IBE-Rule: this rule considers the implied back edge as a set of NRs, maps them to a set of inequations, add the inequations to the \mathcal{L}_E , and determines potential variables that can represent the IBE through elimination of the non-related variables. Assume that for a node x a successor y has been created with $\mathcal{L}(\langle x, y \rangle)$. This implies a back edge for

<i>reset</i> -Rule	<p>if $\{(\leq nR), (\geq nR)\} \cap \mathcal{L}(x) \neq \emptyset$ and $\forall v \in V_x : R \notin \alpha(v)$ then set $\mathcal{L}_E(x) := \emptyset$ and for every successor y of x set $\mathcal{L}(\langle x, y \rangle) := \emptyset$ and, if y in not parent of x set $\mathcal{L}(\langle y, x \rangle) := \emptyset$</p>
<i>reset</i> _{IBE} -Rule	<p>if $Inv(R) \in \mathcal{L}(\langle y, x \rangle)$ but $R \notin \mathcal{L}(\langle x, y \rangle)$ then set $\mathcal{L}_E(x) := \mathcal{L}_E(x) \cup \{\zeta(\mathcal{L}(\langle x, y \rangle), card(y))\}$ and, for every successor y of x set $\mathcal{L}(\langle x, y \rangle) := \emptyset$ and, if y is not parent of x set $\mathcal{L}(\langle y, x \rangle) := \emptyset$</p>
<i>merge</i> -Rule	<p>if there exist root nodes z_a, z_b, z_c for $a, b, c \in I_A$ such that $R' \sqsubseteq_* R_{ab}, R' \in \mathcal{L}(\langle z_a, z_c \rangle)$ then merge the node z_b, z_c and their labels and, replace every occurrence of z_b in the completion graph by z_c</p>
\sqcap -Rule	<p>if $(C_1 \sqcap C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$</p>
\sqcup -Rule	<p>if $(C_1 \sqcup C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{X\}$ for some $X \in \{C_1, C_2\}$</p>
\forall -Rule	<p>if $\forall S.C \in \mathcal{L}(x)$ and there is an S-neighbour y of x with $C \notin \mathcal{L}(y)$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$</p>
$\forall \setminus$ -Rule	<p>if $\forall R \setminus S.C \in \mathcal{L}(x)$ and there is an R-neighbour y of x with $C \notin \mathcal{L}(y)$ and y is not S-neighbour of x then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$</p>
\forall_+ -Rule	<p>if $\forall S.C \in \mathcal{L}(x)$ and there is some R with $Trans(R)$ and $R \sqsubseteq_* S$ and there is R-neighbour y of x with $\forall R.C \notin \mathcal{L}(y)$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$</p>
<i>ch</i> -Rule	<p>if there occurs v in $\mathcal{L}_E(x)$ with $\{v \geq 1, v \leq 0\} \cap \mathcal{L}_E(x) = \emptyset$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{X\}$ for some $X \in \{v \geq 1, v \leq 0\}$</p>
\geq -Rule	<p>if $(\geq nR) \in \mathcal{L}(x)$ and $\xi(R, \geq, n) \notin \mathcal{L}_E(x)$ then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\xi(R, \geq, n)\}$</p>
\leq -Rule	<p>if $(\leq nR) \in \mathcal{L}(x)$ and $\xi(R, \leq, n) \notin \mathcal{L}_E(x)$ then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\xi(R, \leq, n)\}$</p>
<i>IBE</i> -Rule	<p>if $\mathcal{L}_I(x) \neq \emptyset$ and $\{\zeta(\mathcal{L}_I(x), 1)\} \cap \mathcal{L}_E(x) = \emptyset$ then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\zeta(\mathcal{L}_I(x), 1)\}$</p>
<i>BE</i> -Rule	<p>if there exists v occurring in $\mathcal{L}_E(x)$ such that $\sigma(v) = 1, R \in \alpha(v),$ $R \in \mathcal{L}_I(x)$ and y is parent of x with $\mathcal{L}(\langle x, y \rangle) = \emptyset$ then set $\mathcal{L}(\langle x, y \rangle) := \alpha(v)$</p>
<i>RE</i> -Rule	<p>if there exists v occurring in $\mathcal{L}_E(z_a)$ such that $\sigma(v) = 1, z_a, z_b$ root nodes, $R_{ab} \in \alpha(v)$ with $x, b \in I_A$ and $\mathcal{L}(\langle z_a, z_b \rangle) = \emptyset$ then set $\mathcal{L}(\langle z_a, z_b \rangle) := \alpha(v), \mathcal{L}_I(z_b) := Inv(\alpha(v))$</p>
<i>fil</i> -Rule	<p>if there exists v occurring in $\mathcal{L}_E(x)$ such that $\sigma(v) = n$ with $n > 0,$ x is not blocked and $\neg \exists y : \mathcal{L}(\langle x, y \rangle) = \alpha(v)$ then create a new node y and set $\mathcal{L}(\langle x, y \rangle) := \alpha(v),$ $\mathcal{L}_I(y) := Inv(\alpha(v))$ and $card(y) = n$</p>

Fig. 2. The complete tableaux expansion rules for *SHIQ*-ABox

y with a label $\mathcal{L}_I(y) = \text{Inv}(\mathcal{L}(\langle x, y \rangle))$. This back edge is considered as a set of NRs of the form $\geq 1R_i, \leq 1R_i$ where $R_i \in \text{Inv}(\mathcal{L}(\langle x, y \rangle))$. The *IBE-Rule* transforms the implied back edge into a set of inequations in $\mathcal{L}_E(y)$ of the form $(\sum_{v_j \in \mathcal{V}^{R_i}} v_j) \geq 1$ and $(\sum_{v_j \in \mathcal{V}^{R_i}} v_j) \leq 1$ using the function ζ . Since the inequations representing the back edge are restricted to the value one, only one common variable v_k in these inequations will be $\sigma(v_k) = 1$. In addition, ζ ensures that the potential variables for IBE include all the roles in $\mathcal{L}_I(y)$ (see Def. 9).

reset_{IBE}-Rule: this rule extends \mathcal{L}_E as follows. If for a node y and its parent node x , $\mathcal{L}(\langle x, y \rangle) \neq \text{Inv}(\mathcal{L}(\langle y, x \rangle))$, then it implies that a new role should be considered in $\mathcal{L}_E(x)$ due to the restrictions of its child y . Therefore, the *reset_{IBE}-Rule* fires for x where ζ extends $\mathcal{L}_E(x)$ to consider $\text{Inv}(\mathcal{L}(\langle y, x \rangle))$ and ensures that the specific variable representing it is included in the solution as in Def. 8.

reset-Rule: if a new number restriction with a new role R is added to the logical label of a node x , all its children are discarded from the tree and $\mathcal{L}_E(x) = \emptyset$.

BE-Rule: this rule fills a label of the back edge a node to its parent due to the solution of the inequation solver. If a variable v in a solution exists such that $\sigma(v) = 1$ and $\mathcal{L}_I(y) \subseteq \alpha(v)$, then v represents the back edge and the *BE-Rule* fires and fills the edge label.

We adjust the edges between a pair of nodes to satisfy the nature of the inverse roles between them. Interactions of *IBE-Rule*, *BE-Rule*, and *reset_{IBE}-Rule* maintain this characteristic.

RE-Rule: this rule sets the edge between two root nodes. For nodes $a, b \in I_A$, $(a, b) : R$ is considered as $a : \geq 1R_{ab}, a : \leq 1R_{ab}, b : \geq 1\text{Inv}(R_{ab}), b : \leq 1\text{Inv}(R_{ab})$ with $R_{ab} \sqsubseteq R$ and $\text{Inv}(R_{ab}) \sqsubseteq \text{Inv}(R)$. Therefore, a variable with the value of 1, $\sigma(v) = 1$, for node a that contains R_{ab} represents this edge. The *RE-Rule* fires and fills the edge label.

merge-Rule: the *merge-Rule* merges root nodes. Assume three root nodes $a, b, c \in I_A$ where b, c are respectively R -successor and S -successor of a . These assertions will be translated such that we have $a : \geq 1R_{ab}, a : \leq 1R_{ab}, a : \geq 1S_{ac}, a : \leq 1S_{ac}$ with $R_{ab} \sqsubseteq R$ and $S_{ac} \sqsubseteq S$. If there exists a variable v in an arithmetic solution of node a with $R_{ab}, S_{ac} \in \alpha(v)$, it means that c and b need to be merged. The *merge-Rule* merges b and c and w.l.o.g replaces every occurrence of b with c and all outgoing/incoming edges of b become outgoing/incoming edges of c .

ch-Rule: this rule is necessary to ensure the completeness of the algorithm. The partitions of the atomic decomposition represent all possible combinations of the successors of a particular node. The inequation solver has no knowledge about logical reasons that can force a partition to be empty. Thus, from a semantic branching point of view we need to distinguish between the two cases $v \leq 0$ and $v \geq 1$, where v denotes the cardinality of a partition.

fil-Rule: the *fil-Rule* has the lowest priority among the completion rules. This rule is the only one that generates new nodes, called proxy nodes. For example, if a solution Ω for a node x includes a variable v with $\sigma(v) = 2$, the *fil-Rule* creates a proxy node y with cardinality of 2 and sets the edge label and $\mathcal{L}_I(y)$ as shown in Fig. 2. Since this rule generates proxy nodes based on an arithmetic solution that satisfies all the inequations, there is no need to merge the generated proxy nodes later.

The algorithm preserves role hierarchies in its pre-processing phase. If there occurs a variable $v \in \mathcal{L}_E(x)$, where $R \in \alpha(v)$ and $S \notin \alpha(v)$ and $R \sqsubseteq_* S$, then $v \leq 0$. Therefore, the variables that violate the role hierarchy are set to zero. Due to the space limitations, we refer for the proof to [12].

4 Practical Reasoning

In this section, we discuss the complexity of our algorithm and evaluate its behavior in practical reasoning. Let k be the number of all roles occurring in NRs in a TBox after pre-processing and transforming all QNRs to unqualified ones considering inverse roles. The search space of the hybrid algorithm depends on the number of variables occurring in \mathcal{L}_E labels. Since there are k roles, the number of partitions and their associated variables is bounded by $2^k - 1$. The *ch*-Rule creates two branches for each variable: $v \geq 1$ or $v \leq 0$. Consequently, 2^{2^k} cases could be examined by the inequation solver and the worst-case complexity of the algorithm is double exponential. Moreover, the Simplex method which is used in the hybrid algorithm is NP in the worst case. However, in [11] it is shown that integer programming is in P in the worst case, if the number of variables is bounded. The implemented inequation solver (using Simplex method presented in [1]) minimizes the sum of all variables occurring in the inequations. In addition, we use several optimization techniques and heuristics that can eliminate branches in the search space, therefore, avoiding unnecessary invocations of the *ch*-Rule. These techniques dramatically improve the average complexity of the hybrid algorithm over the worst case of 2^{2^k} .

4.1 Optimization Techniques

In most cases, in order to utilize these theoretical algorithms in practice, optimization techniques are required. Due to the complexity of the algorithm, achieving a good performance may seem infeasible. However, optimization techniques can dramatically decrease the size of the search space by pruning many branches. For instance, a standard technique such as axiom absorption [9] often improves reasoning by reducing the number disjunctions. Here we explain some of the optimization techniques used in our hybrid algorithm.

Variable initialization: the inequation solver starts with the default $v \leq 0$ for all variables and later sets some to $v \geq 1$ to satisfy the inequations according to at-least restrictions. Since the *ch*-Rule is invoked 2^{2^k} times in the worst case to check variables for $v \leq 0$ or $v \geq 1$, default zero setting of variables prevents unnecessary invocations of the *ch*-Rule. Moreover, the boundary value of some of the variables can be determined from the beginning according to the occurrence of the numbers in at-most or/and at-least restrictions. For example, if a variable occurs in an at-least restriction but not in an at-most restriction, then it does not have any arithmetic restriction and is called a *don't care* variable [5]. In addition, the *reset_{IBE}*-Rule specifically determines the value of a single variable and sets some of the rest to zero. Similarly, the *IBE*-Rule sets the value of some variables to zero and enforces some of the others to be the potential choices for the answer, therefore, reducing the solution space. Moreover, in order to avoid unnecessary applications of the *reset_{IBE}*-Rule additional heuristics are applied. For a node x , if a role occurs in an at-least restriction but not in any at-most restriction

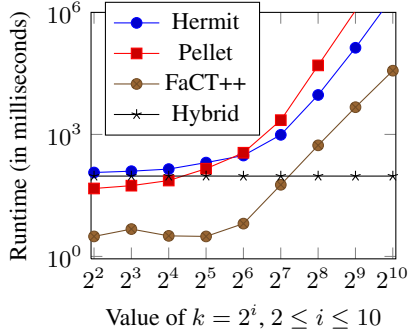


Fig. 3. Runtimes for satisfiable concept $Test_1$ (log-log scale; timeout of 10^6 msecs)

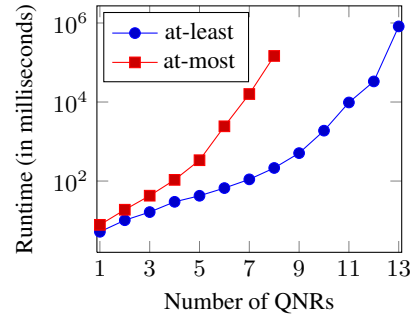


Fig. 4. Runtimes for concept $Test_2$ (using a log scale for the y-axis)

and it is not a sub-role of any role R in a concept of the form $\forall(R \setminus S).C \in \mathcal{L}(x)$, then it cannot be in $\alpha(v)$ where v represents a back edge for node x .

Dependency-directed backtracking: we use backtracking to find sources of logical clashes and then consider the cause of a clash in setting the boundaries of variables in new solutions. This results in pruning branches which would lead to the same clash. If a clash occurs in node x , which was created due to a variable v with $\sigma(v) = k$ and $k \in \mathbb{N}, k \geq 1$, then v must be set to zero. This is called *simple backtracking*. The previous technique can be improved: if a logical clash is encountered due to $\{A, \neg A\} \in \mathcal{L}(x)$, then the source for propagation of these two concepts to $\mathcal{L}(x)$ could be the roles occurring in \forall or $\forall \setminus$ constructs. In this case, the variables which contain all these roles are set to zero. This is called *complex backtracking*. These techniques eliminate many branches in the search space and consequently improving the average complexity of the algorithm.

4.2 A First Evaluation Using Synthetic Benchmarks

In order to evaluate our hybrid algorithm, a prototype reasoner has been implemented in Java using the Web Ontology Language API. The reasoner decides satisfiability of $ALCHIQ$ concepts.

We show the performance of the hybrid reasoner by a set of three benchmarks. Fig. 3 demonstrates the performance of different DL reasoners for testing the satisfiability of the concept $Test_1$ defined as $C \sqcap \geq 2kR \sqcap \forall R. (\geq kR^- . C \sqcap \leq kR^- . C)$, where $k = 2^i, 2 \leq i \leq 10$. Fig. 3 compares the reasoning time of our hybrid reasoner with Pellet, FaCT++, and Hermit.¹ All reasoners determine the satisfiability of the concept in less than ~ 100 ms before reaching $k = 2^4$. While our hybrid reasoner maintains its reasoning time (constantly under 100 ms), Pellet, Hermit, and FaCT++ start exhibiting exponential growth in the reasoning time for values higher than $k = 2^4$. For example, for $k = 2^9$ Pellet's reasoning time is more than 18 minutes and for $k = 2^{10}$ it did not finish within the time limit of 1000 seconds. This example demonstrates the independent behavior of our hybrid calculus in the presence of higher values in QNRs interacting with inverse roles.

¹ We used an AMD 3.4GHz quad core CPU with 16 GB of RAM.

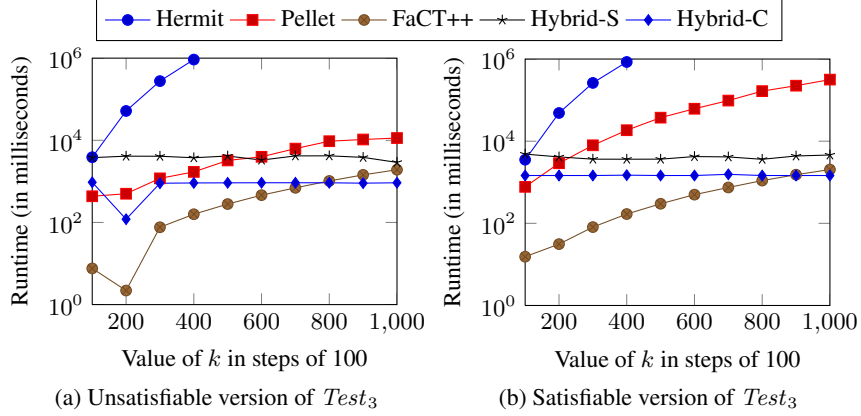


Fig. 5. Linear increase of k (using a log scale for the y-axis)

The second test defines concept $Test_2$ as $\geq 1S.A \sqcap \forall S. \geq 1R.B \sqcap \forall S.\forall R.\forall R^-.P$ with P defined as $\{\sqcap \bowtie 1M_i.C_i \mid 1 \leq i \leq k\}$. Fig. 4 shows the effect of increasing the number of at-least and at-most restrictions in reasoning for $Test_2$. In a model for the concept $Test_2$, the concept expression P is propagated back and will be added to the label of a node which already has $\geq 1R.B$, therefore, we have $(k + 1)$ QNRs. Since for each node which has a parent, an IBE will be considered as a set of two inequations in the \mathcal{L}_E of the node. As shown in Fig. 4, increasing the number of QNRs decreases the performance of the hybrid reasoner. This is due to the fact that a large number of roles in the QNRs increases the number of variables and the size of the search space. Comparing the two diagrams in Fig. 4a and 4b shows that by increasing the number of at-most QNRs the reasoning time for the arithmetic reasoner increases faster than for at-least restrictions. The reason is the heuristic that we explained in Section 4.1. By means of this heuristic, if a role occurs in an at-least restriction and not in any at-most restriction and satisfies the pre-conditions that are mentioned in Section 4.1, then the potential variables for IBE which contain this role are set to zero. Therefore, the number of variables in the search space is decreased.

For the third benchmark we consider the concept $Test_3$ defined as $\geq 8S.A \sqcap \leq 9S.A \sqcap \forall S. (\geq kR.C \sqcap \leq 6T.D \sqcap \geq 5T.D) \sqcap \forall S.\forall R. (\geq 2T.D \sqcap \leq 3T.D) \sqcap \forall S.\forall R.\forall T.\forall T^-. \forall R^-.P$ with $\{R \sqsubseteq M\} \in \mathcal{R}$ and $k \in \{100, \dots, 1000\}$. Fig. 5a displays the runtimes for $Test_3$, where P is defined as $\leq (k - 2)M.(C \sqcup D)$. In this example P is propagated back and causes the unsatisfiability of $Test_3$. Fig. 5b shows the runtimes for $Test_3$ where P is defined as $\leq (k + 1)M.(C \sqcup D)$. In this example P is also propagated back but does not result in the unsatisfiability of $Test_3$. As expected the hybrid reasoner remains stable while the execution times of the other reasoners increase according to the values occurring in the QNRs. As shown in Fig. 5a and 5b, Hermit's behavior is the worst among all the reasoners. Hermit and Pellet show a rapid exponential growth in their reasoning times as a function of k . For $k > 400$, Hermit did not finish within the time limit of 1000 seconds. FaCT++ solves the problem in a more reasonable time, however, it demonstrates its dependency on the value of k as its runtime increases. In addition, Fig. 5 shows the behavior of our hybrid algorithm using *simple* (Hybrid-S)

and *complex* (Hybrid-C) dependency-directed backtracking. The complex backtracking outperforms simple backtracking since it prunes more branches that would lead to the same sort of clash. In addition to improving the performance of the reasoner the optimization techniques used in hybrid reasoner can make its performance more stable.

5 Conclusion and Future Work

The hybrid calculus presented in this paper decides *SHIQ* ABox satisfiability. The implemented prototype demonstrates the improvement on reasoning time for selected benchmarks featuring QNRs and inverse roles. Utilizing algebraic reasoning and applying optimization techniques, the hybrid calculus would be a good solution in case of large numbers occurring in QNRs. In [2] a hybrid algorithm for *SHOQ* using algebraic reasoning has been proposed and extensively analyzed in an empirical evaluation. Due to the nature of nominals, [2] needs to use a global partitioning in contrast to the local partition used in our approach. Several novel techniques to optimize reasoning for *SHOQ* have been designed and evaluated in [2]. Some of these techniques are not exclusively dedicated to nominals and could be applied to our hybrid calculus for *SHIQ* too. For instance, the exponential number of partitions was addressed using a so-called lazy partitioning technique which only generates partitions and their associated variables on demand. We are currently developing a new hybrid prototype for *SHIQ* that integrates suitable optimizations techniques from [2]. We are planning to combine our work on both *SHIQ* and *SHOQ* in order to design a hybrid algebraic calculus for *SHOIQ*.

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, 2nd edn. (Sep 2001)
2. Faddoul, J.: Reasoning Algebraically with Description Logics. Ph.D. thesis, Concordia University (September 2011)
3. Faddoul, J., Farsinia, N., Haarslev, V., Möller, R.: A hybrid tableau algorithm for *ALCQ*. In: Proceedings of ECAI 2008, Greece. pp. 725–726 (2008)
4. Faddoul, J., Haarslev, V.: Algebraic tableau reasoning for the description logic *SHOQ*. Journal of Applied Logic 8(4), 334–355 (December 2010)
5. Farsiniamarj, N., Haarslev, V.: Practical reasoning with qualified number restrictions: A hybrid Abox calculus for the description logic *SHQ*. AI Commun. 23, 205–240 (2010)
6. Haarslev, V., Möller, R.: Optimizing reasoning in description logics with qualified number restriction. In: Proc. of DL 2001, USA. pp. 142–151 (Aug 2001)
7. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. Journal of Logic and Computation 9(3), 385–410 (1999)
8. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic *SHIQ*. In: Proc. of CADE-17. pp. 482–496. Springer-Verlag, London, UK (2000)
9. Horrocks, I., Tobies, S.: Reasoning with axioms: Theory and practice. In: In Proc. of KR 2000. pp. 285–296 (2000)
10. Ohlbach, H.J., Koehler, J.: Modal logics, description logics and arithmetic reasoning. Artif. Intell. 109, 1–31 (1999)
11. Papadimitriou, C.H.: On the complexity of integer programming. J. ACM 28, 765–768 (1981)
12. Roosta Pour, L.: A Hybrid ABox calculus using algebraic reasoning for the description logic *SHIQ*. Master’s thesis, Concordia University (January 2012)