A Saturation-based Algebraic Reasoner for $\mathcal{ELQ}$

Jelena Vlasenko, Maryam Daryalal, Volker Haarslev, and Brigitte Jaumard
Concordia University, Montreal, Quebec, Canada

Abstract

We present a reasoning architecture for deciding subsumption for the description logic $\mathcal{ELQ}$. Our architecture combines saturation rules with algebraic reasoning based on Integer Linear Programming (ILP). Deciding the so-called numerical satisfiability of a set of qualified cardinality restrictions is reduced to constructing a corresponding system of linear inequalities and applying ILP methods in order to determine whether this system is feasible. Our preliminary experiments indicate that this calculus offers a better scalability for qualified cardinality restrictions than approaches combining both tableaux and ILP as well as traditional (hyper)tableau methods.

1 Introduction and Motivation

We present a reasoning architecture for the description logic (DL) $\mathcal{ELQ}$ that is inspired by saturation-based algorithms for $\mathcal{EL}$ [BBL05] but incorporates a new algebraic approach to decide subsumption for $\mathcal{ELQ}$. $\mathcal{ELQ}$ is as expressive as $\mathcal{ALCQ}$ because every $\mathcal{ALCQ}$ Tbox $\mathcal{T}$ can be rewritten to an $\mathcal{ELQ}$ Tbox $\mathcal{T}'$ such that $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$. This is possible since $\mathcal{ELQ}$ allows one to write axioms denoting disjunction and negation via QCRs (see Section 3 for an example).

The performance of the original $\mathcal{ALCQ}$ tableau algorithm [HB91] that is implemented by most DL reasoners covering QCRs is not optimal. To perform a concept satisfiability test, this tableau algorithm creates role successors to satisfy at-least restrictions, e.g. $\geq 20\, R.C$. Given at-most restrictions, e.g., $\leq 10\, R.D$, $\leq 10\, R.E$, the algorithm resolves each R-successor as either D or $\neg D$, and E or $\neg E$. If an at-most restriction for R is violated ($\leq 10\, R.D$), the algorithm nondeterministically merges two R-successors that are instances of D. This uninformed process is highly inefficient, especially when the algorithm has to deal with larger cardinalities and/or large sets of related QCRs. In our previous work (originally inspired by [OK99]) we have shown that algebraic tableau can improve reasoning on QCRs dramatically for DLs such as $\mathcal{SHQ}$ [FH10c], $\mathcal{SHIQ}$ [RH12], and $\mathcal{SHOQ}$ [FH10a, FH10b]. The basic idea in these calculi is to transform a set of QCRs into a system of linear inequalities to be solved by Integer Linear Programming (ILP). If ILP finds a solution to the system of inequalities, i.e., the system is feasible, then the corresponding set of QCRs is satisfiable provided completion rules encounter no logical clashes for the returned numerical solution.

Although the prototypes implementing the above-mentioned approaches on algebraic tableaux [FH10c, RH12, FH10b] could demonstrate runtime improvements of several orders of magnitude for reasoning about QCRs (and nominals) we identified the following disadvantageous characteristics: (i) Given n QCRs (and nominals) the naive encoding of the corresponding system of inequalities requires n rows and 2^m columns, where m is the cardinality of the set P of all the role names and concepts occurring in the n given QCRs. Let us illustrate this with a small example: $\geq 2R.C \sqcap \geq 2R.D \sqcap \leq 2R.E$. In this case, $P = \{R, C, D, E\}$, $n = 3$, $m = 4$. In

In: P. Fontaine, S. Schulz, J. Urban (eds.): Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning (PAAR 2016), Coimbra, Portugal, 02-07-2016, published at http://ceur-ws.org

order to represent the QCRs as inequalities we create $\sum x_{C_i} \geq 2$, $\sum x_{D_j} \geq 2$, $\sum x_{D_k} \geq 2$, and $\sum x_{E_l} \leq 2$. For instance, the variables x_{C_i} represent the cardinalities of all elements in the power set of P that contain C, R. The same holds for the other variables respectively. As an additional constraint we specify that all variables must be greater or equal to zero. Our objective function minimizes the sum of all variables. Intuitively speaking, the above-mentioned concept conjunction is feasible and in this trivial case also satisfiable if the given system of inequalities has an integer solution. It is easy to see that the size of such an inequality system is exponential to m. Furthermore, in order to ensure completeness, we required a so-called choose rule that implements a semantic split that nondeterministically adds for each variable x either the inequality $x \leq 0$ or $x \geq 1$. Unfortunately, this uninformed choose rule could fire 2^{2^m} times in the worst case and cause a severe performance degradation.

(ii) The employed ILP algorithms were best-case exponential in the number of occurring QCRs due to the explicit representation of 2^m variables. In [FH10a, FH10b] we developed an optimization technique called lazy partitioning that tries to delay the creation of ILP variables but it cannot avoid the creation of 2^m variables in case m QCRs are part of a concept model. Our experiments in [FH10a, FH10b, FH10c] indicated that quite a few ILP solutions can cause clashes due to lack of knowledge about known subsumptions, disjointness, and unsatisfiability of concept conjunctions. This knowledge can help reducing the number of variables and eliminating ILP solutions that would fail logically. For instance, an ILP solution for the example presented in the previous paragraph might require the creation of an R-successor as an instance of $C \sqcap D \sqcap \neg E$. However, if C and D are disjoint this ILP solution will cause a clash (and fail logically).

Characteristic (i) can be avoided by eliminating the choose rule for variables. This does not sacrifice completeness because the algorithms implementing our ILP component are complete (and certainly sound) for deciding (in)feasibility. In case a system is feasible (or numerically satisfiable), dedicated saturation rules determine whether the returned solutions are logically satisfiable. In case of logical unsatisfiability a corresponding unsatisfiable concept conjunction is added to the input of the ILP component and therefore monotonically constrains the remaining feasibility space. Consequently, previously computed solutions that result in unsatisfiability are eliminated. For instance, the example above would be deemed as infeasible once the ILP component learns that C and D are subsumed by E and that C and D are disjoint.

The avoidance of characteristic (ii) is motivated by the observation that only a small number of the 2^m variables will have non-zero values in the optimal solution of the linear relaxation, i.e., no more variables than the number of constraints following the characteristics of the optimal solution of a linear program, see, e.g., [Chv83]. As a consequence, only a limited number of variables have a nonzero value in the integer optimal solution. In addition, linear programming techniques such as column generation [DW60, GG61] can operate with as few variables as the set of so-called basic variables in linear programming techniques at each iteration, i.e., nonbasic variables can be eliminated and are not required for the guarantee of reaching the conclusion that a system of linear inequalities is infeasible, or for reaching an optimal LP solution. Although the required number of iterations varies from one case to another, it is usually extremely limited in practice, in the order of few times the number of constraints. The efficiency of the branch-and-price approach, which is required in order to derive an ILP solution, see, e.g., [BJN+98, Van11, LD05], depends on the quality of the integrality gap (i.e., how far the optimal linear programming solution is from the optimal ILP solution in case the system of inequalities is feasible, and on the level of infeasibility otherwise). Several studies have been devoted to improving the convergence of column generation algorithms, especially when degeneracy occurs (see, e.g., [DGL14]) or when convergence is too slow (see, e.g., [ADF09]), and to accelerating the convergence of branch-and-price algorithms [DDS11]. Furthermore, our ILP component considers known subsumptions, disjointness, and unsatisfiability of concept conjunctions and uses a different encoding of inequalities that already incorporates the semantics of universal restrictions. We delegate the generation of inequalities completely to the ILP component.

To summarize, the novel features of our architecture are (i) saturation rules that do not backtrack to decide subsumption (and disjointness); (ii) feasibility of QCRs is decided by ILP (in contrast to [BMC+16]); (iii) our revised encoding of inequalities and the aggregation of information about subsumption, disjointness, and unsatisfiability of concept conjunctions allows a more informed mapping of QCR satisfiability to feasibility and reduces the number of returned solutions that fail logically; (iv) the best-case time complexity of our ILP feasibility test is polynomial to the number of inequalities [Meg87]. In the following sections we present our saturation rules, introduce our ILP approach, sketch soundness, completeness, and termination, present our preliminary evaluation results, and conclude with a summary and future work.

2 A Calculus for $\mathcal{ELQ}$

The DL $\mathcal{ELQ}$ allows the concept-forming constructors $\sqcap$ (conjunction), $\leq$ (at-most restriction), and $\geq$ (at-least restriction). The semantics of $\mathcal{ELQ}$ concepts and roles is defined by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that maps a concept $A \in \mathsf{N_C}$ ($\mathsf{N_C}$ is a set of concept names) to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and a role $R \in \mathsf{N_R}$ ($\mathsf{N_R}$ is a set of roles) to $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We assume the predefined concepts $\top$ and $\bot$ with $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$. $\mathcal{ELQ}$ concepts are inductively defined from the sets $\mathsf{N_C}$ and $\mathsf{N_R}$ using the constructors as follows ($n, m \in \mathbb{N}$, $n \geq 1$, $\|\cdot\|$ denotes set cardinality, $F^{R,C}(x) = \{y \in C^{\mathcal{I}} \,|\, (x,y) \in R^{\mathcal{I}}\}$): (i) $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$; (ii) $(\geq n\, R.C)^{\mathcal{I}} = \{x \,|\, \|F^{C,R}(x)\| \geq n\}$; (iii) $(\leq m\, R.C)^{\mathcal{I}} = \{x \,|\, \|F^{C,R}(x)\| \leq m\}$. The latter two constructors are called QCRs. A concept C is satisfiable if there exists an $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.

An $\mathcal{ELQ}$ Tbox $\mathcal{T}$ is defined as a finite set of axioms of the form $C \sqsubseteq D$ and such an axiom is satisfied by $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We call $\mathcal{I}$ a model of $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$. We use $\mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$ to denote the set that contains $\top$ and all concept names used in $\mathcal{T}$, $\mathsf{C}^{\neg\mathsf{A}}_{\mathcal{T}} = \{\neg A \,|\, A \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}} \setminus \{\top\}\}$, $\mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}$ to denote the set that contains all QCRs (and their negation) used in $\mathcal{T}$, and define $\mathsf{C}_{\mathcal{T}} = \mathsf{C}^{\mathsf{A}}_{\mathcal{T}} \cup \mathsf{C}^{\neg\mathsf{A}}_{\mathcal{T}} \cup \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}$. We use $\mathsf{R}_{\mathcal{T}}$ to denote the set of roles used in $\mathcal{T}$.

A Tbox $\mathcal{T}$ is *normalized* if all axioms are of the form $A_1 \sqsubseteq B$, $A_1 \sqsubseteq \bowtie n\, R.A_2$, $\bowtie n\, R.A_1 \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, with $B \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}} \cup \{\bot\}$, $A_1, A_2 \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$, and $\bowtie \in \{\geq, \leq\}$. It is easy to see that this can be done in polynomial time following the normalizations described in [BBL05, Kaz09, SMH14]. A normalized Tbox is always in negation normal form because it does not allow the occurrence of negation. Some of our saturation rules make use of the negation operator $\dot{\neg}$ which returns the negation normal form for named concepts and QCRs defined as $\dot{\neg}(C) = \neg C$, $\dot{\neg}(\neg C) = C$, $\dot{\neg}(\leq m\, R.C) = \geq m{+}1\, R.C$, and $\dot{\neg}(\geq n\, R.C) = \leq n{-}1\, R.C$ for $m \geq 0$, $n \geq 1$.

We adopted the idea of saturation graphs from [BBL05, SGL14]. Our saturation graphs have been extended to deal with $\mathcal{ELQ}$ and delegate numerical reasoning on QCRs to an ILP component.

A *saturation graph* $G = (V, E, L, L_P, L_P^{\neg})$ is a directed graph where $V = V_P \cup V_A \cup V_C$ such that V_P, V_A, V_C are pairwise disjoint. $V_P = \{v_A \,|\, A \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}\}$ is the set of *predefined* nodes, $V_A = \{v_S \,|\, S \subseteq \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}\}$ the set of *anonymous* nodes representing role successors, and V_C is the set of nodes called *QCR clones* representing subsumption or disjointness tests based on QCRs. Each node $v_S \in V$ uniquely represents either an element ($S \in L(v_S)$) of $\mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$ or a subset ($S \subseteq L(v_S)$) of $\mathsf{C}^{\mathsf{A}}_{\mathcal{T}} \cup \mathsf{C}^{\neg\mathsf{A}}_{\mathcal{T}}$. We also call S the initial label of v_S. Each $v \in V$ is labelled with a set of concepts $L \subseteq \mathsf{C}_{\mathcal{T}} \cup \{\bot\}$ and sets of subsumption tuples $L_P, L_P^{\neg} \subseteq \{\langle p, q\rangle \,|\, p \subseteq \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}, q \subseteq \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}\}$. Each edge $\langle v, v'\rangle \in E$ is labelled with a set $L(v, v') \subseteq \mathsf{R}_{\mathcal{T}}$ where v' is called an r-successor (R-successor) of v with $r \subseteq L(v, v')$ ($R \in L(v, v')$).

Given a normalized Tbox $\mathcal{T}$ and a corresponding saturation graph G the label set L_P ($L_P^{\neg}$) contains tuples specifying subsumption (disjointness) conditions caused by QCRs. A subsumption tuple of the form $\langle p, q\rangle$ consists of a set of precondition concepts $p \subseteq \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$ and a set of QCRs $q \subseteq \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}$. Some of the rules that operate on L_P and $L_P^{\neg}$ make use of special (negated) membership ($\lneqq$, is_new), subset ($\subsetneq$), union ($\subsetneqq$) operators, and a map (φ) operator. We define $\text{is\_new}(q, S) = \langle\{\top\}, \{q\}\rangle \notin S$ and $\text{add\_to}(Q, S)$ as $S \leftarrow S \subsetneqq \{\langle\{\top\}, \{Q\}\rangle\}$ if Q is a QCR or $S \leftarrow S \subsetneqq Q$ if Q is a set of subsumption tuples.

Given sets T, T' of subsumption tuples, $\langle p, q\rangle \lneqq T$ is true if there are p', q' such that $\langle p', q'\rangle \in T$, $p \subseteq p'$, and $q \subseteq q'$; $T \subsetneq T'$ is true if $\forall t \in T \Rightarrow t \lneqq T'$ holds. $\{\langle p_1, s_1\rangle\} \subsetneqq \{\langle p_2, s_2\rangle\}$ is defined as $\{\langle p_2, s_1 \cup s_2\rangle\}$ if $p_1 \subseteq p_2$ and $\{\langle p_1, s_1\rangle, \langle p_2, s_2\rangle\}$ otherwise. The map operator is defined as $\varphi(A, T) = \{\langle p \cup \{A\}, q\rangle \,|\, \langle p, q\rangle \in T\}$ with $A \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$.

Our saturation rules interface the ILP component via the predicate infeasible(S) which transforms all QCRs in S to a corresponding system of inequalities and returns true if this system is infeasible. Otherwise it returns false. If S contains no QCRs it also returns false. A node v is called *numerically satisfiable* or *feasible*, if the set $Q \subseteq L(v)$ of QCRs is feasible. A concept A is called *feasible* if its node v_A is feasible. The number of possible solutions for a feasible node can be huge[1] and quite a few solutions might eventually cause logical unsatisfiability. In order to ensure termination and logically constrain the number of possible solutions for a node v as much as possible, the ILP component makes use of concept information derived from a Tbox $\mathcal{T}$ and its saturation graph G. For a node $v \in V_P \cup V_A$ we define $Q_v = \{C \,|\, \bowtie n\, R.C \in L(v)\} \cup \{\neg D \,|\, \leq 0\, R.D \in L(v)\}$ as the set of QCR qualifications occurring in $L(v)$. The information about concept unsatisfiability and subsumption, which is used by the ILP component, is defined by the set $\mathit{unsat} = \bigcup_{v \in V_P \cup V_A} \mathit{unsat}_v$, which contains sets of concepts whose conjunction is unsatisfiable, and the set $\mathit{subs}_v = \bigcup_{v \in V_P \cup V_A} \mathit{subs}_v$, which contains sets of tuples representing entailed subsumption relationships of the form $A \sqsubseteq B$ or $A_1 \sqcap A_2 \sqsubseteq B$.

$$\mathit{unsat}_v = \{\{A\} \,|\, A \in Q_v, \bot \in L(v_A)\} \cup \{q \,|\, q \subseteq Q_v, \bot \in L(v_q), v_q \in V_A\} \cup \{\{A_i, A_j\} \,|\, \neg A_i \in L(v_{A_j}), \{A_i, A_j\} \cap Q_v \neq \emptyset, i, j \in 1..2, i \neq j\}$$

[1] In [FH10c, Sec. 4.1.1] it was shown that if a node label contains p (q) QCRs of the form $\geq n_i\, R_i.C_i$ ($\leq m_j\, R'_j.C'_j$), then 2^{qN} cases/solutions can exist with $N = \sum_{i=1}^{p} n_i$.

$subs_v = \{\langle\{A_1, A_2\}, B\rangle \mid \{A_1, A_2\} \subseteq Q_v, C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}, C_i \in L(v_{A_i}), B \in L(v_D), i \in 1..2\} \cup$
$\{\langle\{A\}, B\rangle \mid B \in L(v_A), A \in Q_v\}$

In case a node v is numerically satisfiable, the set $\sigma(v)$ contains solutions for feasible inequality systems denoted by the QCRs in $L(v)$. A QCR solution is described by a set of tuples of the form $\langle r, q, n\rangle$ with $r \subseteq \mathsf{R}_\mathcal{T}$ and $q \subseteq \mathsf{C}^{\mathsf{A}}_\mathcal{T} \cup \mathsf{C}^{\neg\mathsf{A}}_\mathcal{T}$ sets of roles and concepts respectively, and n denotes the cardinality of the set of r-successors that are instances of all concepts contained in q.

A saturation graph G is initialized with representative nodes for all concepts in $\mathsf{C}^{\mathsf{A}}_\mathcal{T}$. For instance, for a concept A a node v_A is created with $L(v_A) = \{\top, A\}$ and $L_P(v_A) = L^{\neg}_P(v_A) = \emptyset$. The saturation rules are applied to an initialized saturation graph until no rule is applicable anymore. Only the $\sqsubseteq_{P_4}$-Rule and $\sqsubseteq^{\neg}_{P_4}$-Rule with $v \in V_C$ can be applied to a node v if $\bot \in L(v)$.

This algorithm computes the named subsumers for all predefined nodes and, thus, decides satisfiability and subsumption for all concepts in $\mathsf{C}^{\mathsf{A}}_\mathcal{T}$. A concept A is satisfiable iff $\bot \notin L(v_A)$, and A is subsumed by B iff $B \in L(v_A)$. The top section of Figure 1 contains five rules. The $\sqsubseteq$-Rule ($\sqsubseteq^{\neg}$-Rule) allows a QCR or named concept on its right-hand (left-hand) side. The $\sqsubseteq_*$-Rule ensures that $L(v)$ contains all QCRs and (negated) named concepts inherited from its subsumers. The $\sqsubseteq_\sqcap$-Rule is a standard $\mathcal{EL}$ rule complemented by its contrapositive version ($\sqsubseteq^{\neg}_\sqcap$-Rule). The second section contains two rules that deal with role successors where # denotes the cardinality of an anonymous node. If $\langle r, q, n\rangle \in \sigma(v)$ the fil-rule and e-rule ensure that an edge $r \subseteq L(v, v_q)$ exists that connects v to an anonymous successor v_q with $L(v_q) = q$ and $\#v_q = n$, i.e., v_q represents n identical r-successors of v.

For a node v the $\bot$-Rule records clashes by adding $\bot$ to $L(v)$. For a node v' created by the fil-rule there is no need to propagate $\bot$ to its predecessor v because v' represents one of possibly many existing QCR solutions for v. If all QCR solutions for v fail, the ILP component eventually determines that v is infeasible and thus unsatisfiable.

The fourth section of Figure 1 contains five rules that create and maintain subsumption tuples. Only three of these rules ($P_\bowtie$-Rule, $P^{\neg}_\bowtie$-Rule, $P_\sqcap$-Rule) derive tuples from axioms. The other two rules propagate tuples up ($\sqsubseteq_P$-Rule) or down ($\sqsubseteq^{\neg}_P$-Rule) the subsumption hierarchy.

The bottom section of Figure 1 lists rules that discover subsumptions or disjointness due to subsumption tuples. $Q(v, S)$ is defined as $\bigcup_{\langle p,q\rangle \in S}\{q \mid p \subseteq L(v)\}$ and contains applicable QCRs selected from the set S of subsumption tuples. The relation *clone* $\subseteq V \times (\mathsf{C}^{\mathsf{A}}_\mathcal{T} \cup \mathsf{C}^{\neg\mathsf{A}}_\mathcal{T}) \times V$ keeps track of subsumption or disjointness tests and their associated QCR clones. For instance, a QCR clone $v' \in V_C$ of v_A contains all QCRs contained in $L(v_A)$ and $L_P(v_B)$ ($L^{\neg}_P(v_B)$) and represents the test whether A is subsumed by (disjoint to) B. If v' becomes unsatisfiable, then B ($\neg B$) is added to $L(v_A)$. The $\sqsubseteq_{P_1}$-Rule creates necessary QCR clones, the $\sqsubseteq_{P_2}$-Rule ($\sqsubseteq_{P_3}$-Rule) ensures that $Q(v_A, L_P(v_B))$ ($L(v_A) \cap \mathsf{C}^{\mathsf{Q}}_\mathcal{T}$) is contained in $L(v')$. The $\sqsubseteq_{P_4}$-Rule adds B to $L(v_A)$ if v_A's corresponding clone is unsatisfiable. The other four rules deal with disjointness in an analogous way.

3 Small Tbox Example Illustrating Rule Application

We demonstrate our calculus with a small $\mathcal{ALCQ}$ Tbox $\mathcal{T}$ which entails $C \sqsubseteq D_1 \sqcap \neg D_2$ (see below). A slightly different example (C-SAT-exp-ELQ) is also used in our synthetic benchmark (see Section 6) and surprisingly none of the tested reasoners besides Avalanche and Racer can classify the version with $n = 10$ within a time limit of 1000 seconds.

$C \sqsubseteq \geq 20\, R.\top \sqcap \leq 10\, R.A \sqcap \leq 10\, R.B$
$C \sqsubseteq D_1 \sqcup D_2$
$D_1 \sqsubseteq \leq 10\, R.\neg A$, $D_2 \sqsubseteq \leq 9\, R.\neg B$

The following steps show our rewriting of $\mathcal{T}$ into $\mathcal{T}'$, which is a normalized $\mathcal{ELQ}$ Tbox that is a conservative extension of $\mathcal{T}$.

1. We remove the occurrence of $\sqcup$ and add the entailment $C \sqsubseteq D_1 \sqcup D_2$ using fresh concepts X_3, X_4 and a fresh role S_1.
   $C \sqsubseteq \geq 20\, R.\top \sqcap \leq 10\, R.A \sqcap \leq 10\, R.B$
   $C \sqcap \geq 1\, S_1.(X_3 \sqcap X_4) \sqsubseteq D_1$
   $C \sqcap \geq 2\, S_1.\top \sqsubseteq D_2$
   } *These two axioms entail* $C \sqsubseteq D_1 \sqcup D_2$
   $D_1 \sqsubseteq \leq 10\, R.\neg A$
   $D_2 \sqsubseteq \leq 9\, R.\neg B$

2. We normalize the axioms (but allow a conjunction on the right-hand side for sake of brevity) and replace $\neg A, \neg B$ by $notA, notB$ and new axioms using fresh roles S_2, S_3 such that $\neg notA \sqsubseteq A$ and $\neg notB \sqsubseteq B$ are

$\sqsubseteq$-Rule	**if** $A \sqsubseteq \phi \in \mathcal{T}, \phi \notin L(v_A)$
	then add ϕ to $L(v_A)$
$\sqsubseteq^{\neg}$-Rule	**if** $\phi \sqsubseteq A \in \mathcal{T}, \neg A \in L(v), \dot{\neg}\phi \notin L(v)$
	then add $\dot{\neg}\phi$ to $L(v)$
$\sqsubseteq_*$-Rule	**if** $A \in L(v), \tau \in L(v_A), \tau \notin L(v)$
	then add τ to $L(v)$
$\sqsubseteq_\sqcap$-Rule	**if** $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}, \{A_1, A_2\} \subseteq L(v), B \notin L(v)$
	then add B to $L(v)$
$\sqsubseteq^{\neg}_\sqcap$-Rule	**if** $A_i \sqcap A_j \sqsubseteq B \in \mathcal{T}, \{\neg B, A_i\} \subseteq L(v), \neg A_j \notin L(v)$
	then add $\neg A_j$ to $L(v)$
fil-Rule	**if** $\langle r, q, n\rangle \in \sigma(v), \neg\exists\, v_q \in V : q \subseteq L(v_q), \#v_q \geq n$
	then create $v_q \in V_A$ with $L(v_q) \leftarrow q$ and $\#v_q \leftarrow n$
e-Rule	**if** $\langle r, q, n\rangle \in \sigma(v), q \subseteq L(v_q), \#v_q \geq n, r \notin L(v, v_q)$
	then add r to $L(v, v_q)$
$\bot$-Rule	**if** $\bot \notin L(v) \wedge (\text{infeasible}(L(v)) \vee \{A, \neg A\} \subseteq L(v))$
	then add $\bot$ to $L(v)$
$P_{\bowtie}$-Rule	**if** $\bowtie n\, R.A \sqsubseteq B \in \mathcal{T}, \text{is\_new}(\dot{\neg}(\bowtie n\, R.A), L_P(v_B))$
	then add_to$(\dot{\neg}(\bowtie n\, R.A), L_P(v_B))$
$P^{\neg}_{\bowtie}$-Rule	**if** $A \sqsubseteq \bowtie n\, R.B \in \mathcal{T}, \text{is\_new}(\bowtie n\, R.B, L^{\neg}_P(v_B))$
	then add_to$(\bowtie n\, R.B, L^{\neg}_P(v_B))$
P-Rule	**if** $B \in L(v_A), L_P(v_A) \not\subseteq L_P(v_B)$
	then add_to$(L_P(v_A), L_P(v_B))$
$P^{\neg}$-Rule	**if** $B \in L(v_A), L^{\neg}_P(v_B) \not\subseteq L^{\neg}_P(v_A)$
	then add_to$(L^{\neg}_P(v_B), L^{\neg}_P(v_A))$
$P_\sqcap$-Rule	**if** $A_i \sqcap A_j \sqsubseteq B \in \mathcal{T}, \varphi(A_i, L_P(v_{A_j})) \not\subseteq L_P(v_B)$
	then add_to$(\varphi(A_i, L_P(v_{A_j})), L_P(v_B))$
$\sqsubseteq_{P_1}$-Rule	**if** $Q(v, L_P(v_B)) \neq \emptyset, \{B, \neg B\} \cap L(v) = \emptyset, \mathit{clone}(v, B) = \emptyset$
	then create $v' \in V_C, L(v') \leftarrow \{\top\}, \mathit{clone}(v, B) \leftarrow \{v'\}$
$\sqsubseteq_{P_2}$-Rule	**if** $v' \in \mathit{clone}(v, B), \{B, \neg B\} \cap L(v) = \emptyset, Q(v, L_P(v_B)) \not\subseteq L(v')$
	then add_to$(Q(v, L_P(v_B)), L(v'))$
$\sqsubseteq_{P_3}$-Rule	**if** $v' \in \mathit{clone}(v, B), \{B, \neg B\} \cap L(v) = \emptyset, L(v) \cap \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}} \not\subseteq L(v') \cap \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}$
	then add_to$(L(v) \cap \mathsf{C}^{\mathsf{Q}}, L(v'))$
$\sqsubseteq_{P_4}$-Rule	**if** $\{B, \neg B\} \cap L(v) = \emptyset, v' \in \mathit{clone}(v, B), \bot \in L(v')$
	then add B to $L(v)$
$\sqsubseteq^{\neg}_{P_1}$-Rule	**if** $Q(v, L^{\neg}_P(v_B)) \neq \emptyset, \{B, \neg B\} \cap L(v) = \emptyset, \mathit{clone}(v, \neg B) = \emptyset$
	then create $v' \in V_C, L(v') \leftarrow \{\top\}, \mathit{clone}(v, \neg B) \leftarrow \{v'\}$
$\sqsubseteq^{\neg}_{P_2}$-Rule	**if** $v' \in \mathit{clone}(v, \neg B), \{B, \neg B\} \cap L(v) = \emptyset, Q(v, L^{\neg}_P(v_B)) \not\subseteq L(v')$
	then add_to$(Q(v, L^{\neg}_P(v_B)), L(v'))$
$\sqsubseteq^{\neg}_{P_3}$-Rule	**if** $v' \in \mathit{clone}(v, \neg B), \{B, \neg B\} \cap L(v) = \emptyset, L(v) \cap \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}} \not\subseteq L(v') \cap \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}$
	then add_to$(L(v) \cap \mathsf{C}^{\mathsf{Q}}, L(v'))$
$\sqsubseteq^{\neg}_{P_4}$-Rule	**if** $\{B, \neg B\} \cap L(v) = \emptyset, v' \in \mathit{clone}(v, \neg B), \bot \in L(v')$
	then add $\neg B$ to $L(v)$

Figure 1: Saturation rules ($\phi \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}} \cup \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}$, $\tau \in \mathsf{C}_{\mathcal{T}}$; $\bowtie \in \{\geq, \leq\}$; $i, j \in 1..2$, $i \neq j$)

entailed. This Tbox $\mathcal{T}'$ additionally entails $C \sqsubseteq X_5 \sqcap \neg X_7$.
$C \sqsubseteq \geq 20\, R.\top \sqcap \leq 10\, R.A \sqcap \leq 10\, R.B$
$C \sqcap X_5 \sqsubseteq D_1$
$\geq 1\, S_1.X_6 \sqsubseteq X_5$
$X_3 \sqcap X_4 \sqsubseteq X_6$
$C \sqcap X_7 \sqsubseteq D_2$
$\geq 2\, S_1.\top \sqsubseteq X_7$
$D_1 \sqsubseteq \leq 10\, R.notA$
$D_2 \sqsubseteq \leq 9\, R.notB$
$\leq 1\, S_2.\top \sqsubseteq notA$, $\geq 1\, S_2.\top \sqsubseteq A$ } *These two axioms entail* $\neg notA \sqsubseteq A$
$\leq 1\, S_3.\top \sqsubseteq notB$, $\geq 1\, S_3.\top \sqsubseteq B$ } *These two axioms entail* $\neg notB \sqsubseteq B$

3. Known subsumptions or unsatisfiability of concept conjunctions are derived from $\mathcal{T}'$.
$subs = \{\langle\{C, X_5\}, D_1\rangle, \langle\{C, X_7\}, D_2\rangle, \langle\{X_3, X_4\}, X_6\rangle\}$
$unsat = \emptyset$

4. After applying the rules $\sqsubseteq$, $P_{\bowtie}$, $P_{\bowtie}^{\neg}$, $P_{\sqcap}$ we get the following node labels (for sake of better readability we denote v_A, where A is a concept name, simply as A when used for the labels $L, L_P, L_P^{\neg}$; anonymous and clone nodes are denoted as v_{a_i} and v_{c_j} respectively).
$L(C) = \{\top, C, \geq 1\, S_1.X_3, \geq 1\, S_1.X_4, \geq 20\, R.\top, \leq 10\, R.A, \leq 10\, R.B\}$
$\sigma(C) = \{\langle\{R\}, \{\neg A, \neg B, \neg notB\}, 20\rangle, \langle\{S_1\}, \{X_3, X_4, X_6\}, 1\rangle\}$
$L_P^{\neg}(C) = \{\langle\{\top\}, \{\geq 1\, S_1.X_3\}\rangle, \langle\{\top\}, \{\geq 1\, S_1.X_4\}\rangle, \langle\{\top\}, \{\geq 20\, R.\top\}\rangle, \langle\{\top\}, \{\leq 10\, R.A\}\rangle, \langle\{\top\}, \{\leq 10\, R.B\}\rangle\}$
$L(D_1) = \{\top, D_1, \leq 10\, R.notA\}$
$L_P^{\neg}(D_1) = \{\langle\{\top\}, \{\leq 10\, R.notA\}\rangle\}$
$L_P(D_1) = \{\langle\{\top, C\}, \{\leq 0\, S_1.X_6\}\rangle$
$L(D_2) = \{\top, D_2, \leq 9\, R.notB\}$
$L_P^{\neg}(D_2) = \{\langle\{\top\}, \{\leq 9\, R.notB\}\rangle\}$
$L_P(D_2) = \{\langle\{\top, C\}, \{\leq 1\, S_1.\top\}\rangle$
$L_P(X_5) = \{\langle\{\top\}, \{\leq 0\, S_1.X_6\}\rangle\}$
$L_P(X_7) = \{\langle\{\top\}, \{\leq 1\, S_1.\top\}\rangle\}$
$L_P(notA) = \{\langle\{\top\}, \{\geq 2\, S_2.\top\}\rangle\}$
$L_P(A) = \{\langle\{\top\}, \{\leq 0\, S_2.\top\}\rangle\}$
$L_P(notB) = \{\langle\{\top\}, \{\geq 2\, S_3.\top\}\rangle\}$
$L_P(B) = \{\langle\{\top\}, \{\leq 0\, S_3.\top\}\rangle\}$

5. After applying the rules $\sqsubseteq_{P_1}^{\neg}$, $\sqsubseteq_{P_2}^{\neg}$, $\sqsubseteq_{P_3}^{\neg}$ (is C disjoint to D_2?) and creating the clone v_{c_1}
$clone(C, \neg D_2) \leftarrow v_{c_1}$
$L(v_{c_1}) = \{\top, \geq 1\, S_1.X_3, \geq 1\, S_1.X_4, \geq 20\, R.\top, \leq 10\, R.A, \leq 10\, R.B, \leq 9\, R.notB\}$

6. After applying the fil- and e-rule for v_{c_1} and the rules $\sqsubseteq^{\neg}$, $\bot$ for v_{a_1}
$\sigma(v_{c_1}) = \{\langle\{R\}, \{\neg A, \neg B, \neg notB\}, 20\rangle, \langle\{S_1\}, \{X_3, X_4, X_6\}, 1\rangle\}$
$L(v_{c_1}, v_{a_1}) = \{R\}$
$L(v_{a_1}) = \{\top, \neg A, \neg B, \neg notB, \leq 0\, S_2.\top, \leq 0\, S_3.\top, \geq 2\, S_3.\top, \bot\}$, $\#v_{a_2} = 20$
$L(v_{c_1}, v_{a_2}) = \{S_1\}$
$L(v_{a_2}) = \{\top, X_3, X_4, X_6\}$, $\#v_{a_2} = 1$
Based on the unsatisfiability of v_{a_1} we add $\{\neg A, \neg B, \neg notB\}$ to *unsat*.

7. After applying the fil- and e-rule for v_{c_1} again, the rules $\sqsubseteq^{\neg}$, $\bot$ for v_{a_3}
$\sigma(v_{c_1}) = \{\langle\{R\}, \{\neg A, \neg B, notB\}, 9\rangle, \langle\{R\}, \{\neg A, B, \neg notB\}, 1\rangle, \langle\{R\}, \{A, \neg B, \neg notB\}, 10\rangle, \langle\{S_1\}, \{X_3, X_4, X_6\}, 1\rangle\}$
$L(v_{c_1}, v_{a_3}) = \{R\}$
$L(v_{a_3}) = \{\top, \neg A, \neg B, notB, \leq 0\, S_2.\top, \leq 0\, S_3.\top\}$, $\#v_{a_3} = 9$
$L(v_{c_1}, v_{a_4}) = \{R\}$
$L(v_{a_4}) = \{\top, \neg A, B, \neg notB, \leq 0\, S_2.\top, notA, \geq 2\, S_3.\top\}$, $\#v_{a_4} = 1$

$L(v_{c_1}, v_{a_5}) = \{R\}$
$L(v_{a_5}) = \{\top, A, \neg B, \neg notB, \leq 0\, S_3.\top, notB, \bot\}$, $\#v_{a_2} = 10$
Based on the unsatisfiability of v_{a_5} we add $\{A, \neg B, \neg notB\}$ to *unsat*. Together with the previous addition to *unsat* we learned that $\{\neg B, \neg notB\}$ is unsatisfiable.

8. The $\bot$-rule adds $\bot$ to $L(v_{c_1})$ because infeasible($L(v_{c_1})$) is true due to $\{\neg B, \neg notB\} \in unsat$.
9. The $\sqsubseteq^{\neg}_{P_4}$-rule adds $\neg D_2$ to $L(C)$.
10. The rules $\sqsubseteq^{\neg}_{\sqcap}$ and $\sqsubseteq^{\neg}$ add $\neg X_7$ and $\leq 1\, S_1.\top$ to $L(C)$.
11. After applying the rules $\sqsubseteq_{P_1}$, $\sqsubseteq_{P_2}$, $\sqsubseteq_{P_3}$ (is C subsumed by X_5?) and creating the clone v_{c_2}
$clone(C, X_5) \leftarrow v_{c_2}$
$L(v_{c_2}) = \{\top, \geq 1\, S_1.X_3, \geq 1\, S_1.X_4, \geq 20\, R.\top, \leq 10\, R.A, \leq 10\, R.B, \leq 1\, S_1.\top, \leq 0\, S_1.X_6\}$
12. The $\bot$-rule adds $\bot$ to $L(v_{c_2})$ because infeasible($L(v_{c_2})$) is true due to role S_1.
13. The $\sqsubseteq_{P_4}$-rule adds X_5 to $L(C)$.
14. The $\sqsubseteq_{\sqcap}$-rule adds D_1 to $L(C)$.

Other rules are still applicable but for sake of brevity we do not illustrate their application. Furthermore, no other concept subsumptions are entailed w.r.t. $\mathcal{T}$.

4 ILP Component

We now present the large-scale optimization framework we have designed for solving the system of QCRs in the DL $\mathcal{ELQ}$. It is based on a decomposition technique, called DantzigWolfe decomposition or more commonly column generation technique [Las70, Chv83], combined with a branch-and-price technique [BJN$^+$98] in order to either detect that the system is infeasible, or to compute an optimal solution of the system.

Given $\mathcal{T}$, $v \in V$ and S containing QCRs, define the qualifying role set of v as $S_Q = \{\alpha(\bowtie nR.C) \mid \bowtie nR.C \in S\} \cup \{C \mid \bowtie nR.C \in S\} \cup \{\top, \bot\}$, where α defines a mapping between a QCR and its associated (proxy) subrole of R that is new in $\mathcal{T}$, e.g., $\alpha(\bowtie nR.C) = R'$ with $R' \sqsubseteq R$ and R' new in $\mathcal{T}$. The role partitioning $\mathcal{P}_{S_Q}$ of a qualifying role set S_Q is defined as the power set of S_Q (without the empty set). Note that a qualifying concept can be a member of a partition only if the partition also contains at least one role. For each partition $p \in \mathcal{P}_{S_Q}$, we define $p^I = \mathit{fil}(v,p) \cap (\Delta^I \setminus \bigcup_{Y \in S_Q \setminus p} \mathit{fil}(v,Y))$ where $\mathit{fil}(v,p) = \bigcap_{e \in p} \mathit{fil}(v,e)$ and $\mathit{fil}(v,e) = e^{\mathcal{I}}$ if e is a concept and $\mathit{fil}(v,e) = \{y^{\mathcal{I}} \mid (v^{\mathcal{I}}, y^{\mathcal{I}}) \in e^{\mathcal{I}}\}$ if e is a role. The semantics of role partitions in $\mathcal{P}_{S_Q}$ is defined in such a way that their interpretations are pairwise disjoint. The absence of a role or concept in a partition implies the presence of its semantic negation. We now define a mapping ξ from S to a linear inequality system as follows:

- $\xi(\geq nR.C) = \{\sum_{q \in \mathcal{P}_{S_Q} \mid R' \in q} x_q \geq n\}$ with $R' = \alpha(\geq nR.C)$,
- $\xi(\leq nR.D) = \{\sum_{q \in \mathcal{P}_{S_Q} \mid R'' \in q} x_q \leq n\} \cup \bigcup_{q \in Q} \{x_q \leq 0\}$, such that $Q = \{p \in \mathcal{P}_{S_Q} \mid D \in p \wedge R'' \notin p\}$, with $R'' = \alpha(\leq nR.D)$.

In order to determine whether $\xi(S) = \bigcup_{\bowtie nR.C \in S} \xi(\bowtie nR.C)$ is feasible using a highly scalable solution, we propose a *column generation ILP* formulation for expressing $\xi(S)$ as a mathematical ILP model, which will be solved using a branch-and-price algorithm [BJN$^+$98].

Given $\mathcal{T}$, $v \in V$, S, S_Q and $\mathcal{P}_{S_Q}$, the ILP model associated with the feasibility problem of $\xi(S)$ can be written as follows:

$$\min \quad \sum_{p \in \mathcal{P}_{S_Q}} \mathrm{COST}_p x_p \tag{1}$$

$$\text{subject to:} \quad \sum_{p \in \mathcal{P}_{S_Q}} a_p^{R'} x_p \geq \underline{\delta}_{R'} \qquad R' \in S_Q^{\geq \mathrm{ROLE}} \tag{2}$$

$$\sum_{p \in \mathcal{P}_{S_Q}} a_p^{R'} x_p \leq \bar{\delta}_{R'} \qquad R' \in S_Q^{\leq \mathrm{ROLE}} \tag{3}$$

$$x_p \in \mathbb{Z}^+ \qquad p \in \mathcal{P}_{S_Q}, \tag{4}$$

where $S_Q^{\geq \text{ROLE}} = \{\alpha(\geq nR.C) \mid \geq nR.C \in S\}$, $S_Q^{\leq \text{ROLE}} = \{\alpha(\leq nR.C) \mid \leq nR.C \in S\}$, $S_Q^{\text{ROLE}} = S_Q^{\geq \text{ROLE}} \cup S_Q^{\leq \text{ROLE}}$, $\underline{\delta}_{R'}$ ($\bar{\delta}_{R'}$) is the cardinality of an at-least (at-most) restriction on a subrole $R' \in S_Q^{\text{ROLE}}$, and x_p represents a partition of the set $\mathcal{P}_{S_Q}$. COST_p is defined as the number of concepts in partition p.

When the set of inequalities has a non empty solution set, we are interested in partitions that only contain the entailed concepts, i.e., the minimum number of concepts that are needed to satisfy all the axioms. Consequently, in the objective, COST_p ensures that output partitions only contain the entailed concepts, i.e., the minimum number of concepts that are needed to satisfy all the axioms.

In order to design a scalable solution of (1)-(4), which has an exponential number of variables, we consider a solution scheme that makes use of an *implicit* representation of the constraint matrix. This corresponds to using a branch-and-price method, a generalization of a branch-and-bound method [NW88] in which the linear relaxation of (1)-(4) is solved with a column generation algorithm ([Las70], [Chv83]). The linear relaxation is defined by (1)-(3) and constraint (4′), which is written as follows:

$$x_p \in \mathbb{R}^+ \qquad p \in \mathcal{P}_{S_Q}. \tag{4'}$$

Next step is to use an *implicit* representation of all the variables of (1)-(4′). Therefore, instead of defining the variables for $p \in \mathcal{P}_{S_Q}$, we do it for $p \in \mathcal{P}'_{S_Q} \subseteq \mathcal{P}_{S_Q}$, leading to the so-called restricted ILP or restricted master problem in the mathematical programming literature, in which constraints (4′) are replaced by constraints

$$x_p \in \mathbb{R}^+ \qquad p \in \mathcal{P}'_{S_Q}, \tag{4''}$$

with $\mathcal{P}'_{S_Q} = \emptyset$ at the outset. We next have an iteration solution process in which, at each iteration, the column generation algorithm tries to generate an *improving* partition p, i.e., a partition such that, if added to the current $\mathcal{P}'_{S_Q}$, improves the objective of formulation (1)-(4″). The *partition generator* (PG), also called pricing problem in the mathematical programming literature, is a mathematical ILP model with two sets of decision variables: $a^{R'} = 1$ if role R' is in the generated partition, 0 otherwise ; $b_C = 1$ if concept $C \in S_Q^{\text{CONCEPT}}$ is in the generated partition, 0 otherwise, where S_Q^{CONCEPT} containing all the concepts of S_Q and no subrole. The (PG) model can be stated as:

$$\text{(PG)} \qquad \min \sum_{C \in S_Q^{\text{CONCEPT}}} b_C - \sum_{R' \in S_Q^{\geq \text{ROLE}}} a^{R'} \hat{\pi}^{R'} - \sum_{R' \in S_Q^{\leq \text{ROLE}}} a^{R'} \hat{\omega}^{R'} \tag{5}$$

$$\text{subject to:} \quad a^{R'} \leq b_C \qquad R' \in S_Q^{\geq \text{ROLE}}, C = R'.qualifier \tag{6}$$

$$b_C \leq a^{R'} \qquad R' \in S_Q^{\leq \text{ROLE}}, C = R'.qualifier \tag{7}$$

$$b_C \leq b_\top \qquad C \in S_Q^{\text{CONCEPT}} \tag{8}$$

$$b_\bot = 0 \tag{9}$$

$$b_C, a^{R'} \in \{0,1\} \quad R' \in S_Q^{\text{ROLE}}, C \in S_Q^{\text{CONCEPT}} = S_Q \setminus S_Q^{\text{ROLE}}, \tag{10}$$

where $\hat{\pi}^{R'}$ and $\hat{\omega}^{R'}$ are the optimal values of the dual variables of problem (1)-(4″) (see [Chv83] if not familiar with linear programming concepts). The objective function (5) is the so-called reduced cost of a partition (or column, see again [Chv83] for more details on the column generation method). Constraints (6)-(10) implement the semantics of $\xi(S)$.

(PG) returns a valid partition of P_{S_Q}, assuming no other information is provided. In its feasibility space, it contains a 1-to-1 mapping between the said partitions and algebraic inequalities. If other informations such as: *(i)* subsumption ($\mathcal{T} \models A \sqsubseteq B$), *(ii)* binary subsumption ($A \sqcap B \sqsubseteq C$) and (iii) disjointness ($\mathcal{T} \models A_1 \sqcap \cdots \sqcap A_n \sqsubseteq \bot, n \geq 1$) are provided, the cardinality of some partitions $p \in P_{S_Q}$ might need to be 0. Depending on the information received, the semantics can be implemented by (in)equalities, using a 1-to-1 mapping presented below.

- For every $\mathcal{T} \models A \sqsubseteq B$, set $S_Q^{\text{CONCEPT}} = S_Q^{\text{CONCEPT}} \cup \{A, B\}$ and add $b_A \leq b_B$ to (PG), in order to guarantee that if a generated partition contains a certain concept, it does contain all its subsumers.
- For every $\mathcal{T} \models A \sqcap B \sqsubseteq C$, update $S_Q^{\text{CONCEPT}} = S_Q^{\text{CONCEPT}} \cup \{A, B, C\}$ and add $b_A + b_B - 1 \leq b_C$ to (PG). Then, if (PG) generates a partition containing A and B, it must contain C as well.
- For every disjointness relation $\mathcal{T} \models A_1 \sqcap \cdots \sqcap A_n \sqsubseteq \bot, n \geq 1$, set $S_Q^{\text{CONCEPT}} = S_Q^{\text{CONCEPT}} \cup \{A_1, \ldots, A_n\}$ and add $\sum_{i=1}^{n} b_{A_i} - n + 1 \leq b_\bot$ to (PG). This inequality ensures that if all the concepts A_i, $i = 1, \ldots, n$ are present in a partition; this partition cannot be generated since otherwise it would contain $\bot$ as well.

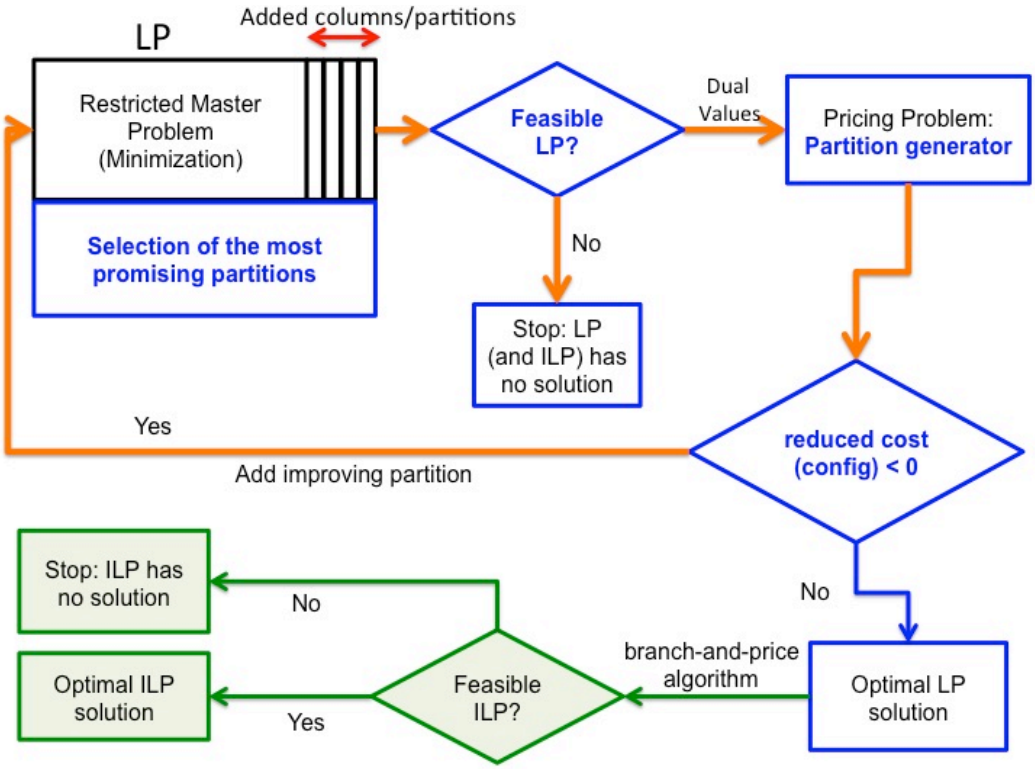

Figure 2: Overview of ILP solution process

- If there is some concept C with $\neg C \in S_Q^{\text{CONCEPT}}$, the relationship of C and $\neg C$ can be modelled with equality $b_C + b_{\neg C} = 1$. This implies that every generated partition contains exactly one of C and $\neg C$.

Every partition generated by (PG), which possibly contains some of the above inequalities or equalities, satisfies all the input axioms except the QCRs, which are taken care of in (1)-(4″). The column generation method is an iterative process in which (1)-(4″) and (PG) are alternatively solved until (PG) cannot provide any new improving partitions, i.e., when the optimal value of the objective of (PG) is positive. This corresponds to one of the optimality conditions when solving a linear program, see, e.g., [Chv83] for more details.

In the branch-and-price algorithm, an implicit enumeration scheme with a breadth first search (BFS) exploration strategy, branching is done upon a subrole R'^* selected as follows:

$$R'^* = \arg \max_{R' \in S_Q^{\text{ROLE}}} \left\{ \sum_{p \in \mathcal{P}'} a_p^{R'} \times \max \{ \hat{x}_p - \lfloor \hat{x}_p \rfloor, \lceil \hat{x}_p \rceil - \hat{x}_p \} \right\}, \tag{11}$$

leading to two branches, one with $a^{R'^*} = 0$ and the other one with $a^{R'^*} = 1$. This branching scheme is clearly exhaustive and valid (see [Van11] and [NW88]). The branch-and-price algorithm terminates when either an integer feasible solution for (1)-(4) is found, or an infeasibility is detected, i.e., either the linear programming relaxation (1)-(4″) has no solution (in most of the infeasibility cases) or the branch-and-price algorithm fails to find a feasible integer solution, see [NW99]. A flowchart is provided in Figure 2 to summarize the solution process.

Soundness and Completeness

The mapping presented in the previous paragraphs is a one-to-one mapping between the axioms and the algebraic inequalities. Thus, the feasibility space defined by (2)-(4) represents all the instances that satisfy all axioms. Following the decomposition ILP model that is proposed, constraints related to satisfying QCRs are addressed in (2)-(4), while all other axioms are embedded in (PG). The branch-and-price algorithm, if (1)-(4) is feasible, returns one set of valid partitions, and the selection of them is made by the objective (1).

The result of the proposed solution process is sound. Indeed, every model produced by the large-scale optimization framework, i.e., ILP model (1)-(4) and branch-and-price algorithm, satisfies all the axioms that are fed to the ILP component. It is also complete, because the feasibility spaces of the (1)-(4″) and (PG) consider all the possible models through the one-to-one mapping that leads to ILP model (1)-(4). In addition, the branching rule in the branch-and-price algorithm considers a partitioning of the overall solution space, therefore no solution is left out.

5 Soundness and Completeness

Due to lack of space we can only sketch the soundness and completeness proofs for our saturation rules. Given the presentation in the previous section we know that the algorithms implementing the ILP component are sound and complete and terminate in finite time for deciding whether a set of QCRs is feasible, i.e., numerically satisfiable.

Intuitively speaking our soundness proof is an extension of the one for $\mathcal{EL}$ but all subsumptions or disjointness involving QCRs are based on QCR infeasibility tests performed by the ILP component. Once a subsumption or disjointness has been proven, its evidence as an element of a node label becomes part of the saturation graph. Since QCRs can interact with one another a QCR solution and its corresponding model is not proof enough for a subsumption unless it is known to be the only possible solution. In order to test QCR-based entailment the label sets L_P and $L_P^{\neg}$ of potential (non-)subsumer nodes together with node labels are used for a feasibility test. If such a set of QCRs has been proven to be infeasible, the corresponding subsumption or disjointness can be added to the graph. Thus, only entailed subsumptions or disjointness will be inferred.

Lemma (Soundness)**.** *Let G be the saturation graph for a normalized Tbox $\mathcal{T}$ after the application of all saturation rules in Figure 1 has terminated and $C \in \mathsf{C}_{\mathcal{T}}^{\mathsf{A}}$ and $D \in \mathsf{C}_{\mathcal{T}}^{\mathsf{A}} \cup \{\bot\}, \beta \in \{D, \doteq\!\neg D\}$. Then $\mathcal{T} \models C \sqsubseteq \beta$ if it holds that $\beta \in L(v_C)$.*

Proof. Let us assume that each rule application creates a new saturation graph and the sequence of rule applications until termination produce a sequence of graphs $G_1, \ldots, G_m$ with $G_i = (V_i, E_i, L_i, L_{Pi}L_{Pi}^{\neg})$ and $i \in 1..m$. Let $F^{R,C}(x) = \{y \in C^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\}$, and $Q(v,S) = \bigcup_{\langle p,q\rangle \in S}\{q \mid p \subseteq L(v)\}$. For all $m \in \mathbb{N}$, models $\mathcal{I}$ of $\mathcal{T}$, $r \subseteq \mathsf{R}_{\mathcal{T}}$, and $x \in C^{\mathcal{I}}$ we claim the following holds: (i) if $\beta \in L_m(v_C)$ then $\mathcal{T} \models C \sqsubseteq \beta$; (ii) if $r \subseteq L_m(v_C, v_q), \#v_q \geq n$ then there exist $y_i \in \Delta^{\mathcal{I}}$ with $(x, y_i) \in \bigcap_{R\in r} R^{\mathcal{I}}, y_i \in \bigcap_{q'\in q} q'^{\mathcal{I}}, i \in 1..n$.

We prove our claim by induction on m. We start with $m = 0$ and assume that $\mathcal{I}$ is a model for $\mathcal{T}$. For G_0 we have $L_0(v_C) = \{\top, C\}$, $L_0(v_C, v_D) = \emptyset$ implying that $D = C$. It is trivial to see that our claim holds. For the induction step we make a case distinction according to applicable rules.

$\sqsubseteq$-Rule Let $\phi \in L_m(v_C) \setminus L_{m-1}(v_C)$ and $x \in C^{\mathcal{I}}$. Then there exists $C \in L_{m-1}(v_C)$ and $C \sqsubseteq \phi \in \mathcal{T}$. Since $\mathcal{I}$ a model of $\mathcal{T}$ it holds $x \in \phi^{\mathcal{I}}$.

$\sqsubseteq^{\neg}$-Rule Let $\doteq\!\neg\phi \in L_m(v) \setminus L_{m-1}(v)$ and $x \in \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. Then there exists $\neg C \in L_{m-1}(v_C)$ and $\phi \sqsubseteq C \in \mathcal{T}$. Since $\mathcal{I}$ a model of $\mathcal{T}$ it holds $x \in \Delta^{\mathcal{I}} \setminus \phi^{\mathcal{I}}$.

$\sqsubseteq_*$-Rule Let $\tau \in L_m(v) \setminus L_{m-1}(v)$ and $x \in C^{\mathcal{I}}$. Then there exists $\tau \in L_{m-1}(v_C)$ and $C^{\mathcal{I}} \subseteq \tau^{\mathcal{I}}$. Since $\mathcal{I}$ a model of $\mathcal{T}$ it holds $x \in \tau^{\mathcal{I}}$.

$\sqsubseteq_\sqcap$-Rule Let $D \in L_m(v) \setminus L_{m-1}(v)$ and $x \in C_1^{\mathcal{I}}$, $x \in C_2^{\mathcal{I}}$. Then there exists $C_1, C_2 \in L_{m-1}(v)$ and $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$. Since $\mathcal{I}$ a model of $\mathcal{T}$ it holds $x \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ and thus $x \in D^{\mathcal{I}}$.

$\sqsubseteq_\sqcap^{\neg}$-Rule Let $\neg C_j \in L_m(v) \setminus L_{m-1}(v)$ and $x \in \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}$, $x \in C_i^{\mathcal{I}}$. Then there exists $\neg D, C_i \in L_{m-1}(v)$ and $C_i \sqcap C_j \sqsubseteq D \in \mathcal{T}$. Since $\mathcal{I}$ a model of $\mathcal{T}$ it holds $x \in C_i^{\mathcal{I}} \cup (\Delta^{\mathcal{I}} \setminus C_j^{\mathcal{I}}) \cup (\Delta^{\mathcal{I}} \setminus D^{\mathcal{I}})$ and thus $x \in \Delta^{\mathcal{I}} \setminus C_j^{\mathcal{I}}$.

fil-Rule, e-Rule Let $v_q \in V_m \setminus V_{m-1}$, $r \subseteq L_m(v_C, v_q)$ such that $q \subseteq L_m(v_q)$ and $\#v_q \geq n$, $Q = L_{m-1}(v_C) \cap \mathsf{C}_{\mathcal{T}}^{\mathsf{Q}}$, and let $x \in \bigcap_{q'\in Q} q'^{\mathcal{I}}$. Due to the soundness and completeness of the ILP component and since $\mathcal{I}$ is a model of $\mathcal{T}$, w.l.o.g. we can assume that that a solution $\langle r, q, n\rangle \in \sigma(v_C)$ exists and we have y_i with $(x, y_i) \in \bigcap_{R\in r} R^{\mathcal{I}}$, $y_i \in \bigcap_{q'\in q} q'^{\mathcal{I}}$ with $i \in 1..n$, and it holds $\forall q' \in Q : x \in q'^{\mathcal{I}}$.

$\bot$-Rule Let $\bot \in L_m(v_C) \setminus L_{m-1}(v_C)$, then either $\{A, \neg A\} \subseteq L_{m-1}(v_C)$ and $x \in A^{\mathcal{I}}, x \in \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$ or infeasible$(L_{m-1}(v_C))$ and by the soundness of the ILP component $x \in \bigcap_{q\in L_{m-1}(v_C)\cap \mathsf{C}_{\mathcal{T}}^{\mathsf{Q}}} q^{\mathcal{I}}$. In both cases it holds that $x \in \bot^{\mathcal{I}}$.

$P_{\bowtie}$**-Rule** Let $\langle\{\top\}, \{\dot{\neg}(\bowtie n\, R.C)\}\rangle \in L_{Pm}(v_D) \setminus L_{Pm-1}(v_D)$. The newly created tuple is correct. Assume for a node v_F that $\bowtie m\, R.E \in L_{m-1}(v_F)$ and $x \in F^{\mathcal{I}}$. If the ILP component determines $\{\bowtie m\, R.E, \dot{\neg}(\bowtie n\, R.C)\}$ as infeasible, then it holds $x \in (\bowtie m\, R.E)^{\mathcal{I}}$, $(\bowtie m\, R.E)^{\mathcal{I}} \subseteq (\bowtie n\, R.C)^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ because $\mathcal{I}$ is a model of $\mathcal{T}$ and $\bowtie n\, R.C \sqsubseteq D \in \mathcal{T}$.

$P^{\neg}_{\bowtie}$**-Rule** Let $\langle\{\top\}, \{\bowtie\ n\, R.C\}\rangle \in L_{Pm}(v_D) \setminus L_{Pm-1}(v_D)$. The newly created tuple is correct. Assume for a node v_F it holds $\bowtie m\, R.E \in L_{m-1}(v_F)$ and $x \in F^{\mathcal{I}}$. If the ILP component determines the set $\{\bowtie m\, R.E, \bowtie n\, R.C\}$ as infeasible, then $x \in (\bowtie m\, R.E)^{\mathcal{I}}$, $\bowtie m\, R.E^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \setminus (\bowtie n\, R.C)^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}$ because $\mathcal{I}$ is a model of $\mathcal{T}$ and $D \sqsubseteq \bowtie n\, R.C \in \mathcal{T}$.

P**-Rule** Let $\langle p, q\rangle \in L_{Pm}(v_D) \setminus L_{Pm-1}(v_D)$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and w.l.o.g. let $\langle p, q\rangle \in L_{Pm-1}(v_C)$. The newly created tuple is correct because $\mathcal{I}$ is a model of $\mathcal{T}$ and a subsumption condition for C is also one for D since $x \in C^{\mathcal{I}}$ implies $x \in D^{\mathcal{I}}$.

$P^{\neg}$**-Rule** Let $\langle p, q\rangle \in L^{\neg}_{Pm}(v_C) \setminus L^{\neg}_{Pm-1}(v_C)$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and w.l.o.g. let $\langle p, q\rangle \in L_{Pm-1}(v_D)$. The newly created tuple is correct because $\mathcal{I}$ is a model of $\mathcal{T}$ and a subsumption condition for $\neg D$ is also one for $\neg C$ since $x \in \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}$ implies $x \in \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$.

$P_{\sqcap}$**-Rule** Let $\langle p \cup \{C_i\}, q\rangle \in L_{Pm}(v_D) \setminus L_{Pm-1}(v_D)$, and w.l.o.g. let $\langle p, q\rangle \in L_{Pm-1}(v_{C_j})$. The newly created tuple is correct because $\mathcal{I}$ is a model of $\mathcal{T}$, $C_i \sqcap C_j \sqsubseteq D \in \mathcal{T}$, and if $C_i^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ then a subsumption condition for C_j is also one for D.

Rules $\sqsubseteq_{P_1}, \sqsubseteq_{P_2}, \sqsubseteq_{P_3}, \sqsubseteq_{P_4}$ Let $\bot \in L_m(v') \setminus L_{m-1}(v')$, $v' \in \mathit{clone}(v_C, D)$, $x \in C^{\mathcal{I}}$. The rule $\sqsubseteq_{P_1}$ added v' to V_{k_1}, the rules $\sqsubseteq_{P_2}, \sqsubseteq_{P_3}$ fired at least once and added the corresponding QCRs to $L_{k_2}(v')$ with $k_1 < k_2 \leq m-1$. We define $Q_P^{\mathcal{I}}(v) = \Delta^{\mathcal{I}} \setminus \bigcap_{q \in Q(v, L_{P_m}(v_D))} q^{\mathcal{I}}$ as the domain complement of all QCRs contained in $Q(v, L_{P_m}(v_D))$ and $Q^{\mathcal{I}}(v) = \bigcap_{q \in L_m(v) \cap \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}} q^{\mathcal{I}}$ as the domain of all QCRs contained in $L_m(v)$. W.l.o.g. we only consider the case that infeasible$(L(v'))$ caused the unsatisfiability of v'. Due to the composition of $L_m(v')$ and since $\mathcal{I}$ is a model of $\mathcal{T}$, it holds $x \in Q^{\mathcal{I}}(v_C)$, $Q^{\mathcal{I}}(v_C) \subseteq Q_P^{\mathcal{I}}(v_C) \subseteq D^{\mathcal{I}}$.

The rules $\sqsubseteq^{\neg}_{P_1}, \sqsubseteq^{\neg}_{P_2}, \sqsubseteq^{\neg}_{P_3}, \sqsubseteq^{\neg}_{P_4}$ are analogous to $\sqsubseteq_{P_1}, \sqsubseteq_{P_2}, \sqsubseteq_{P_3}, \sqsubseteq_{P_4}$ and for lack of space we omit this part of the soundness proof here. □

Lemma (Completeness)**.** *Let G be the saturation graph for a normalized Tbox $\mathcal{T}$ after the application of all saturation rules in Figure 1 has terminated, and let $A \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$, $B \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}} \cup \{\bot\}$, and $\beta \in \{B, \dot{\neg}B\}$. Then, $\beta \in L(v_A)$, if $\mathcal{T} \models A \sqsubseteq \beta$.*

Proof. We show the contrapositive. We assume that if $D \notin L(v_C)$, then $\mathcal{T} \models C \not\sqsubseteq D$. In the following we show that a model for $C \not\sqsubseteq D$ can be derived from the saturation graph G, i.e., by constructing a canonical model $\mathcal{I}(C)$ w.r.t. $\mathcal{T}$ such that $x \in C^{\mathcal{I}} \setminus D^{\mathcal{I}}$. $\mathcal{I}(C)$ is defined iteratively starting with $\mathcal{I}(C_0)$. We define $\Delta^{\mathcal{I}_0(C)} := \{x_C\}$ and $D^{\mathcal{I}_0(C)} := \{x_C \mid D = C\}$ for all $D \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$.

$\sqsubseteq$**-Rule** In case $\phi = D$ then for every $x \in (\Delta^{\mathcal{I}_i} \cap C^{\mathcal{I}_i}) \setminus D^{\mathcal{I}_{i+1}}$ add x to $D^{\mathcal{I}_{i+1}}$. In case $\phi = \bowtie n\, R.C$ the model construction is done in the fil- and e-Rule.

$\sqsubseteq^{\neg}$**-Rule** In case $\phi = D$ then for every $x \in (\Delta^{\mathcal{I}_i} \setminus C^{\mathcal{I}_i}) \setminus (\Delta^{\mathcal{I}_{i+1}} \setminus D^{\mathcal{I}_{i+1}})$ add x to $\Delta^{\mathcal{I}_{i+1}} \setminus D^{\mathcal{I}_{i+1}}$. In case $\phi = \bowtie n\, R.C$ the model construction is done in the fil- and e-Rule item below.

$\sqsubseteq_*$**-Rule** This rule has no impact on the model construction.

$\sqsubseteq_{\sqcap}$**-Rule** For every $x \in (\Delta^{\mathcal{I}_i} \cap C_1^{\mathcal{I}_i} \cap C_2^{\mathcal{I}_i}) \setminus D^{\mathcal{I}_i}$ add x to $D^{\mathcal{I}_{i+1}}$.

$\sqsubseteq^{\neg}_{\sqcap}$**-Rule** For every $x \in (\Delta^{\mathcal{I}_k} \cap C_i^{\mathcal{I}_k}) \setminus (C_j^{\mathcal{I}_k} \cup D^{\mathcal{I}_k})$ add x to $D^{\mathcal{I}_{k+1}}$ $(i, j \in 1..2, i \neq j)$.

fil-Rule, e-Rule Let $r^{\mathcal{I}_i} = \bigcap_{R \in r} R^{\mathcal{I}_i}$, $q^{\mathcal{I}_i} = \bigcap_{q' \in q} q'^{\mathcal{I}_i}$, and $k \in 1..n$. For every $x \in \Delta^{\mathcal{I}_i} \cap C^{\mathcal{I}_i}$ where no tuples $(x, y_k) \in r^{\mathcal{I}_{i+1}}$ with $y_k \in q^{\mathcal{I}_{i+1}}$ exist, introduce new individuals y_k to $\Delta^{\mathcal{I}_{i+1}}$, for every $q' \in q$ add y_k to $q'^{\mathcal{I}_{i+1}}$, and add (x, y_k) to $R^{\mathcal{I}_{i+1}}$ for every $R \in r$.

Rules $\sqsubseteq_{P_1}, \sqsubseteq_{P_2}, \sqsubseteq_{P_3}, \sqsubseteq_{P_4}$ Let $Q_P^{\mathcal{I}_i} = \bigcap_{q \in Q(v, L_P(v_F))} q^{\mathcal{I}_i}$, $Q_{P^{\neg}}^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \setminus Q_P^{\mathcal{I}_i}$, $Q_v^{\mathcal{I}_i} = \bigcap_{q \in L_{v_E} \cap \mathsf{C}^{\mathsf{Q}}_{\mathcal{T}}} q^{\mathcal{I}_i}$, $E, F \in \mathsf{C}^{\mathsf{A}}_{\mathcal{T}}$, and $x \in E^{\mathcal{I}_i}$. By the composition rules for L_P and the infeasibility result of the ILP component we know that $E^{\mathcal{I}_i} \subseteq Q_{P^{\neg}}^{\mathcal{I}_i} \subseteq F^{\mathcal{I}_i}$ because $E^{\mathcal{I}_i} \cap Q_P^{\mathcal{I}_i} = \emptyset$. Now for every $x \in (\Delta^{\mathcal{I}_i} \cap E^{\mathcal{I}_i}) \setminus F^{\mathcal{I}_{i+1}}$ add x to $F^{\mathcal{I}_{i+1}}$.

The rules $\sqsubseteq^{\neg}_{P_1}, \sqsubseteq^{\neg}_{P_2}, \sqsubseteq^{\neg}_{P_3}, \sqsubseteq^{\neg}_{P_4}$ are analogous to $\sqsubseteq_{P_1}, \sqsubseteq_{P_2}, \sqsubseteq_{P_3}, \sqsubseteq_{P_4}$ and for lack of space we omit this part of the completeness proof here.

The construction of a model is performed in a fair manner, i.e., every rule applicable to already existing elements in $\Delta^{\mathcal{I}_i}$ is applied before applying rules to new elements. The constructed model $\mathcal{I}(\mathcal{C})$ is indeed a model of C w.r.t $\mathcal{T}$. Due to lack of space we cannot prove this in detail but we argue informally. Although our rules deal with $\mathcal{ELQ}$, a subset of these deal with $\mathcal{EL}$ if one assumes that only QCRs of the form $\geq 1R.C$ are allowed. Thus, we claim that our rules are complete for $\mathcal{EL}$. Due to the semantics of QCRs and their allowed occurrence on the left- and and right-hand side of axioms, these can entail subsumptions that are not possible in $\mathcal{EL}$. In order to ensure completeness we introduced the labels L_P and $L^{\neg}_P$ that aggregate possible (non-)subsumption conditions caused by QCRs. In order to prove subsumption or disjointness we create corresponding QCR clones. In case a clone is satisfiable after all rules have terminated this clone can serve as a model for the non-subsumption or non-disjointness. In case additional entailed information is added to the saturation graph that causes the unsatisfiability of a QCR clone, we will add the entailed subsumption or disjointness to the sets *subs* and *unsat* that are used by the ILP component. Regarding the fil- and e-rule there could exist another possible incompleteness because a numerically satisfiable set of QCRs could eventually cause the unsatisfiability of a created anonymous node due to entailed information missing in the graph at the time of the feasibility test. However, if such an anonymous node becomes unsatisfiable, the cause of unsatisfiability is learned by adding its cause to the sets *subs* and *unsat*. This ensures that a previously computed solution is monotonically constrained until either an anonymous node is determined as satisfiable or the set of QCRs becomes eventually infeasible. Again, this ensures that no model for non-subsumption or non-disjointness is overlooked and no subsumption or disjointness is missed. □

Lemma (Termination)**.** *For every normalized $\mathcal{ELQ}$ Tbox over $\mathsf{C}_\mathcal{T}$ and $\mathsf{R}_\mathcal{T}$ the application of the saturation rules in Figure 1 will terminate in finite time.*

Proof. The algorithm terminates naturally because the saturation graph is finite and its number of nodes is bounded by the size of the power set of $\mathsf{C}_\mathcal{T}$ and $\mathsf{R}_\mathcal{T}$ respectively, cycles are avoided by reusing nodes in accordance to standard subset blocking. Finally, the number of concepts, edges, QCRs, and subsumption tuples that can be added to L, L_P, and $L^{\neg}_P$ is again bounded by the size of the power set of $\mathsf{C}_\mathcal{T}$ and $\mathsf{R}_\mathcal{T}$ respectively. All rules extend labels monotonically. Once the precondition of a rule is true for elements of a graph G and the rule has fired, it cannot fire again for the same set of elements, thus, any series of rule applications is finite. □

6 A First Experimental Evaluation

We developed a prototype system called Avalanche that implements our calculus as proof of concept.[2] Its ILP component is based on IBM ILOG CPLEX Optimization Studio [IBM] that is freely available for academic research. Avalanche currently parses only $\mathcal{ELQ}$ ontologies expressed in RDF/XML syntax. For this first release our focus is on a sound, complete, and terminating implementation. We did not optimize Avalanche for $\mathcal{EL}$ ontologies yet and many of our saturation rules are implemented very inefficiently. Deciding subsumption for named concepts where L_P and/or $L^{\neg}_P$ are not empty is currently computed very naively. We generate a QCR clone for each possible subsumption or disjointness via QCRs and the number of clones is bounded by n^2 if the Tbox contains n named concepts.

Avalanche has been written in the Java programming language. It consists of two main components: a reasoner and an external ILP module. The ILP module is needed for feasibility testing of QCRs. The reasoner executes in the following way: First, it parses the input ontology file. The axioms in the file must conform to our $\mathcal{ELQ}$ syntax. If an axiom in the ontology does not conform to the $\mathcal{ELQ}$ syntax restrictions an exception will be raised and the system will terminate. After the parsing is completed, the input axioms are rewritten to our internal normal form. Then, we build nodes representing all concepts declared in the original ontology file as well as the ones obtained during the normalization phase.

Our reasoning process is divided into two phases: an unfolding and a rule application phase. During the unfolding phase we apply the rules $\sqsubseteq$, $\sqsubseteq_{\bowtie}$, $\sqsubseteq^{\neg}_{\bowtie}$, P, and $P^{\neg}$. In this phase, each rule is applied only once to each node. Afterwards, during the rule application phase, we apply all rules rules except $\sqsubseteq$, $\sqsubseteq_{\bowtie}$, and $\sqsubseteq^{\neg}_{\bowtie}$. The rules do not have to be applied in any particular order. We apply all matching rules to all the nodes. The system calls the external ILP module when it needs to solve a system of inequalities.

[2] *Avalanche web page*: https://users.encs.concordia.ca/%7Ehaarslev/Avalanche/

Ontology Name	#Axioms	#Concepts	#Roles	#QCRs
canadian-parliament-factions-1	48	21	6	19
canadian-parliament-factions-2	56	24	7	25
canadian-parliament-factions-3	64	27	8	30
canadian-parliament-factions-4	72	30	9	35
canadian-parliament-factions-5	81	34	10	40
canadian-parliament-full-factions-1	51	22	6	22
canadian-parliament-full-factions-2	60	25	7	30
canadian-parliament-full-factions-3	69	28	8	36
canadian-parliament-full-factions-4	78	31	9	42
canadian-parliament-full-factions-5	87	34	10	48
C-SAT-exp-ELQ	26	10	4	13
C-UnSAT-exp-ELQ	26	10	4	13
genomic-cds rules-ELQ-fragment-1	716	358	1	357
genomic-cds rules-ELQ-fragment-2	718	359	1	357

Figure 3: Metrics of $\mathcal{ELQ}$ benchmark ontologies (#=Number of ...)

The experiments were performed on a MacBook Pro (2.6 GHz Intel Core i7 processor, 16GB memory) using only one core. The comparison results (average of 3 runs) are shown in Figure 4. We compared Avalanche with major OWL reasoners: FaCT++ (1.6.4) [FaC], HermiT (1.3.8) [Her], Konclude (0.6.2) [Kon], and Racer (3.0) [HM01, HHMW12, Rac], which is the only other available OWL reasoner using an ILP component for reasoning about QCRs. The algorithms implementing Racer's ILP component are in general best-case exponential to the number of QCRs given for one concept. Some metrics about the benchmark ontologies are shown in Figure 3.

The first benchmark (see top part of Figure 4) uses variants of two real-world ontologies modelling a component of the Canadian Parliament, namely the House of Commons, which is a democratically elected body that had 308 members in the last Parliament (most members elected in 2011) [Can]. For each Canadian province a faction is defined [Can] that is composed of members elected in the same province. In order to ensure subsumptions based on QCRs we defined four concepts categorizing factions as tiny, small, medium, or big, based on the number of faction members. Each faction belongs to exactly one of these faction categories. The second version adds a concept defining for the House of Commons the number of members from each province [Can]. The only reasoners that can classify all variants of the simplest of these ontologies within the given time limit are Avalanche and Racer. Avalanche still exhibits an exponential runtime increase but it is the only reasoner that can classify all variants of these ontologies. The performance of Racer for the simpler ontology is a good indication that Avalanche can achieve similar runtimes with optimized reasoning algorithms.

The second benchmark (see middle part of Figure 4) uses synthetic concept templates that are similar to the example used in Section 3. The original $\mathcal{ALCQ}$ concepts are shown below the table. They were manually rewritten into normalized $\mathcal{ELQ}$ as demonstrated in Section 3. The concept templates use a variable n that is increased exponentially. The numbers used in the template are bounded by the value of $2n$. The first template is satisfiable and the second one unsatisfiable. Only Avalanche and Racer can classify all variants of these small ontologies within the time limit.

The third benchmark (see bottom part of Figure 4) uses two $\mathcal{ELQ}$ fragments of a real world ontology, genomic-cds_rules [Sam13, Sam14], which was developed for pharmacogenetics and clinical decision support. It contains many concepts using QCRs of the form $= 2\, has.A_i$. However, in these fragments the concepts (A_i) occurring in the qualification of the QCRs do not interact with one another. This simplifies the reasoning and all reasoners except Racer perform well. Avalanche and HermiT as well as FaCT++ and Konclude have similar runtimes. These fragments are interesting because the concept `<#human>` contains several hundred QCRs using the same role. This is one of the reasons why Racer timed out for both fragments.

7 Summary

We presented the first calculus that combines saturation and ILP-based reasoning to decide subsumption for $\mathcal{ELQ}$. Our preliminary experiments indicate that this calculus offers a better scalability for qualified cardinality restrictions than approaches combining both tableaux and ILP as well as traditional (hyper)tableau methods. We already started to improve the efficiency of our implementation and expect a significant speedup for our next

Canadian Parliament Factions only						Canadian Parliament Full				
#F	**Ava**	**Fac**	**Her**	**Kon**	**Rac**	**Ava**	**Fac**	**Her**	**Kon**	**Rac**
5	11.1	TO	TO	TO	0.12	375	TO	TO	TO	TO
4	8.4	TO	TO	TO	0.11	22.4	TO	TO	TO	TO
3	2.7	TO	TO	TO	0.07	4.1	TO	TO	TO	TO
2	1.4	TO	TO	TO	0.07	2.7	TO	TO	TO	10.5
1	0.8	TO	TO	7.3	0.05	1.1	TO	TO	TO	0.44

C-SAT-exp-ELQ						C-UnSAT-exp-ELQ				
n	**Ava**	**Fac**	**Her**	**Kon**	**Rac**	**Ava**	**Fac**	**Her**	**Kon**	**Rac**
40	0.69	TO	TO	TO	0.01	0.63	TO	TO	TO	0.01
20	0.62	TO	TO	TO	0.01	0.80	TO	TO	TO	0.01
10	0.63	TO	TO	TO	0.01	0.99	TO	TO	TO	0.01
5	0.72	6.3	4.4	0.91	0.01	0.74	TO	TO	784	0.01
3	0.62	0.17	0.18	0.33	0.01	0.75	0.25	1.15	1.18	0.01

Sat: $C \sqsubseteq (\leq n\, R.\neg A \sqcup \leq n{-}1\, R.\neg B) \sqcap \geq 2n\, R.\top \sqcap \leq n\, R.A \sqcap \leq n\, R.B$
Unsat: $C \sqsubseteq (\leq n{-}1\, R.\neg A \sqcup \leq n{-}1\, R.\neg B) \sqcap \geq 2n\, R.\top \sqcap \leq n\, R.A \sqcap \leq n\, R.B$

Satisfiability of concept `<#human>`					
Name	**Ava**	**Fac**	**Her**	**Kon**	**Rac**
genomic-cds_rules-ELQ-fragment-1	1.13	27.7	0.87	27.7	TO
genomic-cds_rules-ELQ-fragment-2	2.4	28.2	1.14	28.3	TO

Figure 4: Benchmark runtimes in seconds with a timeout of 1000 seconds (TO=timeout, #F=Number of Factions, Ava=Avalanche, Fac=FaCT++, Her=HermiT, Kon=Konclude, Rac=Racer)

release. We plan to add a preprocessing phase that automatically rewrites $\mathcal{ALCQ}$ ontologies to our $\mathcal{ELQ}$ syntax. Our next steps are to extend our calculus to cover role hierarchies and inverse roles.

References

[ADF09] H. M. B. Amor, J. Desrosiers, and A. Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009.

[BBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, pages 364–369, 2005.

[BJN+98] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[BMC+16] A. Bate, B. Motik, B. Cuenca Grau, F. Simančík, and I. Horrocks. Extending consequence-based reasoning to $\mathcal{SRIQ}$. In *Proc. of KR*, pages 187–196, 2016.

[Can] Canadian Parliament. https://en.wikipedia.org/wiki/House_of_Commons_of_Canada.

[Chv83] V. Chvatal. *Linear Programming*. Freeman, 1983.

[DDS11] G. Desaulniers, J. Desrosiers, and S. Spoorendonk. Cutting planes for branch-and-price algorithms. *Networks*, 58:301–310, 2011.

[DGL14] J. Desrosiers, J. Gauthier, and M. Lübbecke. Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, 236:453–460, July 2014.

[DW60] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[FaC] FaCT++. http://owl.cs.manchester.ac.uk/tools/fact/.

[FH10a] J. Faddoul and V. Haarslev. Algebraic tableau reasoning for the description logic $\mathcal{SHOQ}$. *Journal of Applied Logic*, 8(4):334–355, 2010.

[FH10b] J. Faddoul and V. Haarslev. Optimizing algebraic tableau reasoning for $\mathcal{SHOQ}$: First experimental results. In *Proc. of DL*, pages 161–172, 2010.

[FH10c] N. Farsiniamarj and V. Haarslev. Practical reasoning with qualified number restrictions: A hybrid Abox calculus for the description logic $\mathcal{SHQ}$. *AI Communications*, 23(2-3):334–355, 2010.

[GG61] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[HB91] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR*, pages 335–346, 1991.

[Her] HermiT. http://www.hermit-reasoner.com/download.html.

[HHMW12] V. Haarslev, K. Hidde, R. Möller, and M. Wessel. The RacerPro knowledge representation and reasoning system. *Semantic Web*, 3(3):267–277, 2012.

[HM01] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR*, pages 701–705, 2001.

[IBM] IBM. https://www.ibm.com/developerworks/downloads/ws/ilogcplex/.

[Kaz09] Y. Kazakov. Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies. In *Proc. of IJCAI*, pages 2040–2045, 2009.

[Kon] Konclude. http://www.derivo.de/en/produkte/konclude/.

[Las70] L. Lasdon. *Optimization Theory for Large Systems*. MacMillan, New York, 1970.

[LD05] M. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.

[Meg87] N. Megiddo. On the complexity of linear programming. In *Advances in Economic Theory*, pages 225–268. Cambridge University Press, 1987.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

[NW99] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1999. reprint of the 1988 edition.

[OK99] H. Ohlbach and J. Köhler. Modal logics, description logics and arithmetic reasoning. *Artificial Intelligence*, 109(1-2):1–31, 1999.

[Rac] Racer. https://www.ifis.uni-luebeck.de/index.php?id=385.

[RH12] L. Roosta Pour and V. Haarslev. Algebraic reasoning for $\mathcal{SHIQ}$. In *Proc. of DL*, pages 530–540, 2012.

[Sam13] M. Samwald. Genomic CDS: an example of a complex ontology for pharmacogenetics and clinical decision support. In *2nd OWL Reasoner Evaluation Workshop*, pages 128–133. CEUR, 2013.

[Sam14] M. Samwald. An update on genomic CDS, a complex ontology for pharmacogenomics and clinical decision support. In *3rd OWL Reasoner Evaluation Workshop*, pages 58–63. CEUR, 2014.

[SGL14] A. Steigmiller, B. Glimm, and T. Liebig. Coupling tableau algorithms for expressive description logics with completion-based saturation procedures. In *Proc. of IJCAR*, pages 449–463, 2014.

[SMH14] F. Simančík, B. Motik, and I. Horrocks. Consequence-based and fixed-parameter tractable reasoning in description logics. *Artificial Intelligence*, 209:29–77, 2014.

[Van11] F. Vanderbeck. Branching in branch-and-price: a generic scheme. *Mathematical Programming*, 130(2):249–294, 2011.