Handling Nominals and Inverse Roles using Algebraic Reasoning

Humaira Farid and Volker Haarslev

Concordia University, Montreal
h_arid@encs.concordia.ca, haarslev@cse.concordia.ca

Abstract. This paper presents a novel $\mathcal{SHOI}$ tableau calculus which incorporates algebraic reasoning for deciding ontology consistency. Numerical restrictions imposed by nominals, existential and universal restrictions are encoded into a set of linear inequalities. Column generation and branch-and-price algorithms are used to solve these inequalities. Our preliminary experiments indicate that this calculus performs better on $\mathcal{SHOI}$ ontologies than standard tableau methods.

1 Introduction and Motivation

Description Logic (DL) is a formal knowledge representation language that is used for modeling ontologies. Modern description logic systems provide reasoning services that can automatically infer implicit knowledge from explicitly expressed knowledge. Designing reasoning algorithms with high performance has been one of the main concerns of DL researchers. One of the key features of many description logics is support for nominals. Nominals are special concept names that must be interpreted as singleton sets. They allow to use Abox individuals within concept descriptions. However, nominals carry implicit global numerical restrictions that increase reasoning complexity. Moreover, the interaction between nominals and inverse roles leads to the loss of the tree model property. Most state-of-the-art reasoners, such as Konclude [23], Fact++ [24], HermiT [21], have implemented traditional tableau algorithms. Konclude also incorporated consequence-based reasoning into its tableau calculus [22]. These reasoners try to construct completion graphs in a highly non-deterministic way in order to handle nominals. For example, a small $\mathcal{ALCO}$ ontology models Canada consisting of its ten provinces: $CA\_Province \equiv \{Ontario, Quebec, NovaScotia, NewBrunswick, Manitoba, BritishColumbia, PrinceEdwardIsland, Saskatchewan, NewfoundlandAndLabrador, Alberta\}$. If one tries to model that Canada consists of 11 provinces, it is trivial to see that it is not possible because the cardinality of $CA\_Province$ is implicitly restricted to the 10 provinces listed as nominals. However, according to our preliminary experiments, above mentioned DL reasoners are unable to decide this inconsistency within a reasonable amount of time. Consequence-based (CB) reasoning algorithms are also extended to more expressive DLs such as $\mathcal{SHOI}$ [6] and $\mathcal{SROIQ}$ [7]. Since their implementations are not available, we could not analyze these reasoners.

However, algebraic DL reasoners are considered more efficient in handling numerical restrictions [10,13,14,25]. RacerPro [14] was the first highly optimized reasoner that combined tableau-based reasoning with algebraic reasoning [15]. Other tableau-based algebraic reasoner for $\mathcal{SHQ}$ [13], $\mathcal{SHIQ}$ [20], $\mathcal{SHOQ}$ [11,10] are also proposed to handle qualified number restrictions (QNRs) and their interaction with inverse roles or nominals. These reasoners use an atomic decomposition technique to encode number restrictions into a set of linear inequalities. These inequalities are then solved by integer linear programming (ILP). These reasoners perform very efficiently in handling huge values in number restrictions. However, their ILP algorithms are best-case exponential to the number of inequalities. For example, in case of m inequalities they require 2^m variables in order to find the optimal solution. However, for ILP with a huge number of variables it is not feasible to enumerate all variables. To overcome this problem, the column generation technique has been used [25,27] which considers a small subset of variables. However, to the best of our knowledge, no algebraic calculus can handle DLs supporting nominals and inverse roles simultaneously.

In this paper, we present a novel algebraic tableau calculus for $\mathcal{SHOI}$ to handle a large number of nominals and their interaction with inverse roles. The rest of this paper is structured as follows. Section 2 defines important terms and introduces $\mathcal{SHOI}$. Section 3 presents the algebraic tableau calculus for $\mathcal{SHOI}$. Section 4 provides evaluation results for the implemented prototype Cicada. The last section concludes our paper. An extended version of this paper [12] contains more details about ILP and the example presented in Section 3.3.

2 Preliminaries

In this section, we introduce $\mathcal{SHOI}$ and some notations used later. Let $N = N_C \cup N_o$ where N_C represents concept names and N_o nominals. Let N_R be a set of role names with a set of transitive roles $N_{R_+} \subseteq N_R$. The set of roles in $\mathcal{SHOI}$ is $N_R \cup \{R^- \mid R \in N_R\}$ where R^- is called the inverse of R. A function Inv returns the inverse of a role such that $\mathsf{Inv}(R) = R^-$ if $R \in N_R$ and $\mathsf{Inv}(R) = S$ if $R = S^-$ and $S \in N_R$. An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$ of individuals called the domain of interpretation and an interpretation function $\cdot^\mathcal{I}$. Table 1 presents syntax and semantic of $\mathcal{SHOI}$. We use $\top$ ($\bot$) as an abbreviation for $A \sqcup \neg A$ ($A \sqcap \neg A$) for some $A \in N_C$. In the following $\sharp\{.\}$ denotes set cardinality.

A role inclusion axiom (RIA) of the form $R \sqsubseteq S$ is satisfied by $\mathcal{I}$ if $R^\mathcal{I} \subseteq S^\mathcal{I}$. We denote with $\sqsubseteq_*$ the transitive, reflexive closure of $\sqsubseteq$ over N_R. If $R \sqsubseteq_* S$, we call R a subrole of S and S a superrole of R. A general concept inclusion (GCI) $C \sqsubseteq D$ is satisfied by $\mathcal{I}$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$. A role hierarchy $\mathcal{R}$ is a finite set of RIAs. A Tbox $\mathcal{T}$ is a finite set of GCIs. A Tbox $\mathcal{T}$ and its associated role hierarchy $\mathcal{R}$ is satisfied by $\mathcal{I}$ (or consistent) if each GCI and RIA is satisfied by $\mathcal{I}$. Such an interpretation $\mathcal{I}$ is then called a model of $\mathcal{T}$. A concept description C is said to be satisfiable by $\mathcal{I}$ iff $C^\mathcal{I} \neq \emptyset$. An Abox $\mathcal{A}$ is a finite set of assertions of the form $a : C$ (concept assertion) with $a^\mathcal{I} \in C^\mathcal{I}$, and $(a, b) : R$ (role assertion) with $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$. Due to nominals, a concept assertion $a : C$ can be transformed into a concept inclusion $\{a\} \sqsubseteq C$ and a role assertion $(a, b) : R$ into $\{a\} \sqsubseteq \exists R.\{b\}$. Therefore, concept satisfiability and Abox consistency can be reduced to Tbox consistency

Table 1. Syntax and semantics of $\mathcal{SHOI}$

Construct	**Syntax**	**Semantics**
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
exists restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \land y \in C^{\mathcal{I}}\}$
nominal	$\{o\}$	$\sharp\{o\}^{\mathcal{I}} = 1$
role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse role	R^-	$(R^-)^{\mathcal{I}} = \{(y, x) \mid (x, y) \in R^{\mathcal{I}}\}$
transitive role	$R \in N_{R_+}$	$(x, y) \in R^{\mathcal{I}} \land (y, z) \in R^{\mathcal{I}} \Rightarrow (x, z) \in R^{\mathcal{I}}$

by using nominals. We use $\{o_1, \ldots, o_n\}$ as an abbreviation for $\{o_1\} \sqcup \cdots \sqcup \{o_n\}$ and may write $\{o\}$ as o. Moreover, we do not make the unique name assumption; therefore, two nominals might refer to the same individual.

Nominals carry implicit global numerical restrictions. For example, if $C \sqsubseteq \{o_1, o_2, o_3\}$ (or $\{o_1, o_2, o_3\} \sqsubseteq C$), then o_1, o_2, o_3 impose a numerical restriction that there can be at most (or at least, if $o_1, o_2, o_3 \in N_o$ are declared as pairwise disjoint) three instances of C. These restrictions are global because they affect the set of all individuals of C in $\Delta^{\mathcal{I}}$. These implicit numerical restrictions increase reasoning complexity.

3 An Algebraic Tableau Calculus for $\mathcal{SHOI}$

In this section, we present an algebraic tableau calculus for $\mathcal{SHOI}$ that decides Tbox consistency. Since nominals carry numerical restrictions, algebraic reasoning is used to ensure their semantics. The algorithm takes a $\mathcal{SHOI}$ Tbox $\mathcal{T}$ and its role hierarchy $\mathcal{R}$ as input and tries to create a complete and clash-free completion graph in order to check Tbox consistency. The reasoner is divided into two modules: 1) Tableau Module (TM), and 2) Algebraic Module (AM).

Let $G = (V, E, \mathcal{L}, \mathcal{B})$ be a completion graph for a $\mathcal{SHOI}$ Tbox $\mathcal{T}$ where V is a set of nodes and E a set of edges. Each node $x \in V$ is labelled with a set of concepts $\mathcal{L}(x)$, and each edge $\langle x, y\rangle \in E$ with a set of role names $\mathcal{L}(x, y)$. For each node $x \in V$, if $\mathcal{L}(x)$ contains a universal restriction on role R and there exists an R-neighbour of x, then $\mathcal{B}(x)$ contains a tuple of the form $\langle v, \mathcal{L}(x, v)\rangle$ where $v \in V$ is an R-neighbour of x. We use $\sharp v$ to denote the cardinality of a node v. For convenience, we assume that all concept descriptions are in negation normal form.

TM starts with some preprocessing and reduces all the concept axioms in a Tbox $\mathcal{T}$ to a single axiom $\top \sqsubseteq C_{\mathcal{T}}$ such that $C_{\mathcal{T}} := \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} nnf(\neg C \sqcup D)$, where *nnf* transforms a given concept expression to its negation normal form. The algorithm checks consistency of $\mathcal{T}$ by testing the satisfiability of $o \sqsubseteq C_{\mathcal{T}}$ where $o \in N_o$ is a fresh nominal in $\mathcal{T}$, which means that at least $o^{\mathcal{I}} \in {C_{\mathcal{T}}}^{\mathcal{I}}$ and ${C_{\mathcal{T}}}^{\mathcal{I}} \neq \emptyset$. Moreover, since $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ then every domain element must also satisfy $C_{\mathcal{T}}$. For creating a complete and clash-free completion graph, TM

$\sqcap$-Rule	**if** $(C_1 \sqcap C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ **then** set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-Rule	**if** $(C_1 \sqcup C_2) \in \mathcal{L}(x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ **then** set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\forall$-Rule	**if** $\forall S.C \in \mathcal{L}(x)$ and there $\exists y$ with $R \in \mathcal{L}(x, y)$, $C \notin \mathcal{L}(y)$ and $R \sqsubseteq_* S$ **then** set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$
$\forall_+$-Rule	**if** $\forall S.C \in \mathcal{L}(x)$ and there exist U, R with $R \in N_{R_+}$ and $U \sqsubseteq_* R$, $R \sqsubseteq_* S$, and a node y with $U \in \mathcal{L}(x, y)$ and $\forall R.C \notin \mathcal{L}(y)$ **then** set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$
nom_{merge}-Rule	**if** for some $o \in N_o$ there are nodes x, y with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$, $x \neq y$ **then** if x is an initial node, then merge y into x, else merge x into y
inverse-Rule	**if** $\forall R^-.C \in \mathcal{L}(y)$, $R \in \mathcal{L}(x, y)$, and $\langle x, \mathcal{L}(y, x)\rangle \notin \mathcal{B}(y)$ **then** set $\mathcal{B}(y) = \mathcal{B}(y) \cup \{\langle x, \mathcal{L}(y, x)\rangle\}$
fil-Rule	**if** $\langle \mathsf{R}, \mathsf{C}, n, \mathsf{V}\rangle \in \sigma(x)$ and x is not blocked **then** 1. **if** $\mathsf{V} = \emptyset$ and there exists no R-neighbour y of x with $\mathsf{C} \subseteq \mathcal{L}(y)$, $\sharp y \geq n$, **then** create a new node y with $\mathcal{L}(y) \leftarrow \mathsf{C}$ and $\sharp y \leftarrow n$ 2. **else** for all $v \in \mathsf{V}$ add C to $\mathcal{L}(v)$ and set $\sharp v = n$
e-Rule	**if** $\langle \mathsf{R}, \mathsf{C}, n, \mathsf{V}\rangle \in \sigma(x)$ and $\mathsf{C} \subseteq \mathcal{L}(y)$, $\sharp y \geq n$, $\mathsf{R} \not\subseteq \mathcal{L}(x, y)$ **then** merge R into $\mathcal{L}(x, y)$ and $\{\mathsf{Inv}(R) \mid R \in \mathsf{R}\}$ into $\mathcal{L}(y, x)$, and for all S with $R \sqsubseteq_* S \in \mathcal{R}$ add S to $\mathcal{L}(x, y)$ and $\mathsf{Inv}(S)$ to $\mathcal{L}(y, x)$

Fig. 1. The expansion rules for $\mathcal{SHOI}$

applies expansion rules (see Figure 1 and Section 3.1). AM handles all numerical restrictions using ILP. It generates inequalities and solves them using the branch-and-price technique (see Section 3.2 for details). We use equality blocking [18,16] due to the presence of inverse roles.

3.1 Expansion Rules

In order to check the consistency of a Tbox $\mathcal{T}$, the proposed algorithm creates a completion graph G using the expansion rules shown in Figure 1. A node x in G contains a clash if $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for $A \in N_C$ or AM has no feasible solution for x. G is complete if no expansion rule is applicable to any node in G. $\mathcal{T}$ is consistent if G is complete and no node in G contains a clash.

The $\sqcap$-Rule, $\sqcup$-Rule and $\forall$-Rule are similar to standard tableau expansion rules for $\mathcal{ALC}$. The $\forall_+$-Rule preserves the semantics of transitive roles. The nom_{merge}**-Rule** merges two nodes containing in their label the same nominal. Suppose there is $o \in \mathcal{L}(x)$ and $o \in \mathcal{L}(y)$, and nodes x and y are not the same, then nom_{merge}-Rule merges x into y. It adds $\mathcal{L}(x)$ to $\mathcal{L}(y)$ and moves all edges leading to (from) x so that they lead to (from) y. For each node z, if $\langle z, y\rangle \in E$ and $\langle z, x\rangle \in E$, then $\mathcal{L}(z, y) = \mathcal{L}(z, y) \cup \mathcal{L}(z, x)$. Similarly, if $\langle y, z\rangle \in E$ and $\langle x, z\rangle \in E$, then $\mathcal{L}(y, z) = \mathcal{L}(y, z) \cup \mathcal{L}(x, z)$. It also merges $\mathcal{B}(x)$ into $\mathcal{B}(y)$.

If $\mathcal{L}(x, y) = \{R\}$ and $\forall R^-.C \in \mathcal{L}(y)$, then the *inverse***-Rule** encodes for AM the already existing R^--edge by adding a tuple $\langle x, \{R^-\}\rangle$ to $\mathcal{B}(y)$. AM plays also an important role if nominals occur in universal restriction. For example, consider the axioms $A \sqsubseteq \exists R.B$, $B \sqsubseteq \exists R^-.C \sqcap \exists R^-.D \sqcap \forall R^-.\{o_1, o_2\}$ and $o_1 \sqcap o_2 \sqsubseteq \bot$, where $A, B, C, D \in N_C$, $o_1, o_2 \in N_o$ and $R \in N_R$. Suppose we have $A \in \mathcal{L}(x)$, $R \in \mathcal{L}(x, y)$ and $B \in \mathcal{L}(y)$. Since nominals carry numerical restrictions,

$\forall R^{-}.\{o_1, o_2\}$ implies that we can have at most 2 R^{-}-neighbours of y. However, standard tableau reasoners might create two new R^{-}-neighbours of y without considering the existing R^{-}-neighbour x of y. Then they try to merge these three nodes in a non-deterministic way to satisfy the numerical restriction imposed by nominals. In our approach, the *inverse*-Rule encodes information about an existing R^{-}-neighbour of y and AM generates a deterministic solution.

For a node x, AM transforms all existential restrictions, universal restrictions and nominals to a corresponding system of inequalities. AM then processes these inequalities and gives back a solution set $\sigma(x)$. The set $\sigma(x)$ is either empty or contains solutions derived from feasible inequalities. In case of infeasibility AM signals a clash. A solution is defined by a set of tuples of the form $\langle \mathsf{R}, \mathsf{C}, n, \mathsf{V}\rangle$ with $\mathsf{R} \subseteq N_R$, $\mathsf{C} \subseteq N$, $n \in \mathbb{N}$, $n \geq 1$ and $\mathsf{V} \subseteq V$. Each tuple represents n R-neighbours of x (where R is a set of roles) that are instances of all elements of C. Here, V is an optional set that contains existing R-neighbours of x that must be reused and C is added to their labels. Consider the axiom $A \sqsubseteq \exists R.B \sqcap \exists R.C \sqcap \forall S.\{o\}$, where $A, B, C \in N_C$, $o \in N_o$, $R, S \in N_R$, $R \sqsubseteq S$, and $A \in \mathcal{L}(x)$. AM returns the solution $\sigma(x) = \{\langle \{R, S\}, \{B, C, o\}, 1\rangle\}$. The ***fil*-Rule** is used to generate nodes based on the arithmetic solution that satisfies a set of inequalities. For the above solution, the *fil*-Rule creates one node y with cardinality 1 such that $\mathcal{L}(y) \leftarrow \{B, C, o\}$ and $\sharp y = 1$. The ***e*-Rule** creates an edge between nodes x and y, and adds R, S to $\mathcal{L}(x, y)$ and $\mathsf{Inv}(R), \mathsf{Inv}(S)$ to $\mathcal{L}(y, x)$. The *e*-Rule always adds all implied superroles to edge labels.

3.2 Generating Inequalities

Dantzig and Wolfe [8] proposed a column generation technique for solving linear programming (LP) problems, called Dantzig–Wolfe decomposition, where a large LP is decomposed into a master problem and a subproblem (or pricing problem). In case of LP problems with a huge number of variables, column generation works with a small subset of variables and builds a Restricted Master Problem (RMP). The Pricing Problem (PP) generates a new variable with the most reduced cost if added to RMP (see [4,26] for details). However, column generation may not necessarily give an integral solution for an LP relaxation, i.e., at least one variable has not an integer value. Therefore, the branch-and-price method [3] has been used which is a combination of column generation and branch-and-bound technique [9]. We employ this technique by mapping number restrictions to linear inequality systems using a column generation ILP formulation (see [26] for details). CPLEX [5] has been used to solve our ILP formulation.

Encoding Existential Restrictions and Nominals into Inequalities The atomic decomposition technique [19] is used to encode numerical restrictions on concepts and role fillers into inequalities. These inequalities are then solved for deciding the satisfiability of the numerical restrictions. The existential restrictions are converted into ≥ 1 inequalities. The cardinality of a partition element containing a nominal o is equal to 1 due to the nominal semantics; $\sharp\{o\}^{I} = 1$ for each nominal $o \in N_o$. Therefore, the decomposition set is defined as $Q = Q_{\exists} \cup Q_{\forall} \cup Q_o$, where $Q_{\exists}$ ($Q_{\forall}$) contains existential (universal) restrictions

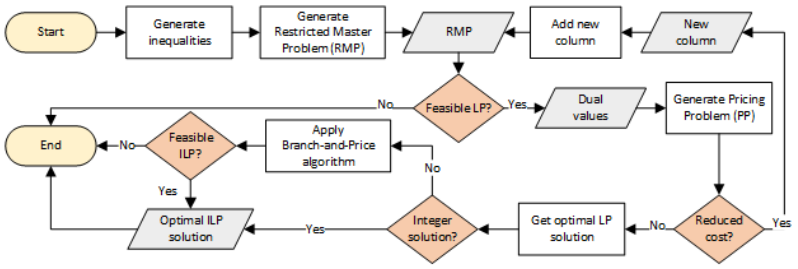

Fig. 2. Overview of the algebraic reasoning process

and Q_o contains all related nominals. Each element $R_q \in Q_\exists \cup Q_\forall$ represents a role $R \in N_R$ and its qualification concept expression q and each element $I_q \in Q_o$ represents a nominal $q \in N_o$. The elements in $Q_\forall$ are used by AM to ensure the semantics of universal restrictions. The set of related nominals $Q_o \subseteq N_o$ is defined as $Q_o = \{o \mid o \in clos(q) \wedge R_q \in Q_\exists \cup Q_\forall\}$ where $clos(q)$ is the closure of concept expression q. The atomic decomposition considers all possible ways to decompose Q into sets that are semantically pairwise disjoint.

Branch-and-Price Method In the following, we use a Tbox $\mathcal{T}$ and its role hierarchy $\mathcal{R}$, a completion graph G, a decomposition set Q and a partitioning $\mathcal{P}$ that is the power set of Q containing all subsets of Q except the empty set. Each partition element $p \in \mathcal{P}$ represents the intersection of its elements. We decompose our problem into two subproblems: (i) restricted master problem (RMP), and (ii) pricing problem (PP). RMP contains a subset of columns and PP computes a column that can maximally reduce the cost of RMP's objective. Whenever a column with negative reduced cost is found, it is added to RMP. Number restrictions are represented in RMP as inequalities, with a restricted set of variables. The flowchart in Figure 2 illustrates the whole process.

Restricted Master Problem RMP is obtained by considering only variables x_p with $p \in \mathcal{P}'$ and $\mathcal{P}' \subseteq \mathcal{P}$ and relaxing the integrality constraints on the x_p variables. Initially $\mathcal{P}'$ is empty and RMP contains only artificial variables h to obtain an initial feasible inequality system. Each artificial variable corresponds to an element in $Q_\exists \cup Q_o$ such that h_{R_q}, $R_q \in Q_\exists$ and h_{I_q}, $I_q \in Q_o$. An arbitrarily large cost M is associated with every artificial variable. If any of these artificial variables exists in the final solution, then the problem is infeasible. The objective of RMP is defined as the sum of all costs as shown in (1) of the RMP below.

$$\text{Min} \sum_{p \in \mathcal{P}'} cost_p x_p + M \sum_{R_q \in Q_\exists} h_{R_q} + M \sum_{I_q \in Q_o} h_{I_q} \quad \text{subject to} \tag{1}$$

$$\sum_{p \in \mathcal{P}'} a_p^{R_q} x_p + h_{R_q} \geq 1 \quad R_q \in Q_\exists \tag{2}$$

$$\sum_{p \in \mathcal{P}'} a_p^{I_q} x_p + h_{I_q} = 1 \quad I_q \in Q_o \tag{3}$$

$$x_p \in \mathbb{R}^+ \text{ with } p \in \mathcal{P}' \tag{4}$$

$$a_p^{R_q}, a_p^{I_q} \in \{0,1\},\ h_{R_q}, h_{I_q} \in \mathbb{R}^+ \text{ with } R_q \in Q_\exists,\ I_q \in Q_o$$

where a decision variable x_p represents the elements of the partition element $p \in \mathcal{P}'$. The coefficients a_p are associated with variables x_p and $a_p^{R_q}$ indicates whether an R-neighbour that is an instance of q exists in p. Similarly, $a_p^{I_q}$ indicates whether a nominal q exists in p. The weight $cost_p$ defines the cost of selecting p and it depends on the number of elements p contains. Since we minimize the objective function, $cost_p$ in the objective (1) ensures that only subsets with entailed concepts will be added which are the minimum number of concepts that are needed to satisfy all the axioms. Constraint (2) encodes existential restrictions and (3) numerical restrictions imposed by nominals (i.e., $\sharp\{o\}^I = 1$). Constraint (4) states the integrality condition relaxed from $x_p \in \mathbb{Z}^+$ to $x_p \in \mathbb{R}^+$.

Pricing Problem: The objective of PP uses the dual values π, ω as coefficients of the variables that are associated with a potential partition element. The binary variables r_{R_q}, r_{I_q}, b_q ($q \in N$) are used to ensure the description logic semantics. A binary variable $r_{R_\top}$ is used to handle role hierarchy. A variable b_q is set to 1 if there exists an instance of concept q and r_{R_q} is set to 1 if there exists an R-neighbour that is an instance of concept q. Likewise, r_{I_q} is set to 1 if there exists a nominal q. Otherwise these variable are set to 0. The PP is given below.

$$\text{Min} \sum_{q \in N} b_q - \sum_{R_q \in Q_\exists} \pi_{R_q} r_{R_q} - \sum_{I_q \in Q_o} \omega_{I_q} r_{I_q} \quad \text{subject to} \tag{5}$$

$$\begin{aligned}
r_{R_q} - b_q &\leq 0 && R_q \in Q_\exists, R \in N_R, q \in N_C && (6)\\
r_{I_q} - b_q &= 0 && I_q \in Q_o, q \in N_o && (7)\\
r_{R_q} - r_{R_\top} &\leq 0 && R \in N_R, q \in N && (8)\\
r_{R_\top} - b_q &\leq 0 && R_q \in Q_\forall, R \in N_R, q \in N && (9)\\
r_{R_\top} - r_{S_\top} &\leq 0 && R \sqsubseteq S \in \mathcal{R}, R, S \in N_R && (10)\\
& b_q, r_{R_q}, r_{I_q}, r_{R_\top}, r_{S_\top} \in \{0,1\}
\end{aligned}$$

where vector π and ω are dual variables associated with (2) and (3) respectively. For each at-least restriction represented in (2), Constraint (6) is added to PP, which ensures that if $r_{R_q} = 1$ then variable for b_q must exist in $\mathcal{P}'$. Similarly, (7) ensures the semantics of nominals represented in (2). Constraints (8) - (10) ensure the semantics of universal restrictions and role hierarchies respectively.

We can also map the semantics of selected DL axioms, where only atomic concepts occur, into inequalities, as shown in Table 2. For every $\mathcal{T} \models A \sqcap B \sqsubseteq C$, AM adds $b_A + b_B - 1 \leq b_C$ to PP. Therefore, if PP generates a partition containing A and B, then it must also contain C. Similarly, for every $\mathcal{T} \models A \sqsubseteq B \sqcup C$, AM adds $b_A \leq b_B + b_C$ to PP. This inequality ensures that if a partition contains A, then it must also contain B or C.

Soundness and Completeness of Algebraic Module All existential restrictions and nominals are converted into linear inequalities and added to RMP.

Table 2. DL axioms and their corresponding PP inequalities ($n \geq 1$)

DL Axiom	Inequality in PP	Description
$A_1 \sqcap ... \sqcap A_n \sqsubseteq B$	$\sum_{i=1}^{n} b_{A_i} - (n-1) \leq b_B$	If a set contains $A_1, ..., A_n$, then it also contains B.*
$A \sqsubseteq B_1 \sqcup ... \sqcup B_n$	$b_A \leq \sum_{i=1}^{n} b_{B_i}$	If a set contains A, then it also contains at least one concept from $B_1, ..., B_n$.

*Encodes unsatisfiability and disjointness in case $B \sqsubseteq_* \bot$

Other axioms, such as universal restrictions, role hierarchy, subsumption and disjointness, are embedded in PP. In case of feasible inequalities, the branch-and-price algorithm returns a solution set that contains valid partition elements. Since the branch-and-price algorithm satisfies all the axioms embedded in RMP and PP, this solution is sound. Moreover, it is also complete because CPLEX is used to solve linear inequalities and it does not overlook any possible solution.

Proposition 1. *For a set of inequalities, the arithmetic module either generates an optimal solution which satisfies all inequalities or detects infeasibility.*

3.3 Example Illustrating Rule Application and ILP formulation

Consider the small Tbox

$$A \sqsubseteq \exists R.B \sqcap \exists R.\{o1\}$$
$$B \sqcap \{o1\} \sqsubseteq \bot$$
$$B \sqsubseteq \exists R.C \sqcap \exists R^{-}.D \sqcap \forall S^{-}.\{o2\}$$
$$C \sqcap D \sqsubseteq \bot$$
$$C \sqsubseteq \exists R.E$$
$$E \sqsubseteq \forall S^{-}.\{o1\}$$

with $N_R = \{R, S\}$, $\{A, B, C, D, E\} \subseteq N_C$, $\{o1, o2\} \subseteq N_o$, and $R \sqsubseteq S \in \mathcal{R}$. For the sake of better readability, we apply in this example lazy unfolding [2,17].

1. We start with root node x and its label $\mathcal{L}(x) = \{A\}$ and by unfolding A and applying the $\sqcap$-Rule we get $\mathcal{L}(x) = \{A, \exists R.B, \exists R.\{o1\}\}$.
2. Since $\{\exists R.B, \exists R.\{o1\}\} \subseteq \mathcal{L}(x)$, AM generates a corresponding set of inequlities and applies ILP considering known subsumption and disjointness.
3. For solving these inequalities, RMP starts with artificial variables, $\mathcal{P}'$ is initially empty, and $Q_\exists = \{R_B, R_{o1}\}$, $Q_\forall = \emptyset$ and $Q_o = \{I_{o1}\}$ (see Fig. 3). The objective of (PP 1a) uses the dual values from (RMP 1a). For each at-least restriction a constraint (e.g., $\exists R.B \rightsquigarrow r_{R_B} - b_B \leq 0$) is added to (PP 1a), which indicates that if $r_{R_B} = 1$ then a variable b_B will also be 1. Constraint (i) ensures that B and $o1$ cannot exist in same partition element. Constraint (ii) ensures the semantics of nominals.
4. The values of $r_{R_{o1}}, r_{I_{o1}}$ are 1 in (PP 1a), therefore, the variable $x_{R_{o1}I_{o1}}$ is added to (RMP 1b). Since only one b variable (i.e., b_{o1}) is 1, the cost of $x_{R_{o1}I_{o1}}$ is 1. $\mathcal{P}' = \{\{R_{o1}, I_{o1}\}\}$ and the value of the objective function is reduced from 30 in (RMP 1a) to 11 in (RMP 1b).

RMP 1a

Min $$10h_{R_B} + 10h_{R_{o1}} + 10h_{I_{o1}}$$

Subject to: $$h_{R_B} \geq 1$$ $$h_{R_{o1}} \geq 1$$ $$h_{I_{o1}} = 1$$

Solution: $cost = 30$, $h_{R_B} = 1$, $h_{R_{o1}} = 1, h_{I_{o1}} = 1$
Duals: $\pi_{R_B} = 10, \pi_{R_{o1}} = 10, \omega_{I_{o1}} = 10$

PP 1a

Min
$$b_B + b_{o1} - 10r_{R_B} - 10r_{R_{o1}} - 10r_{I_{o1}}$$

Subject to:

$$r_{R_B} - b_B \leq 0 \qquad b_B + b_{o1} \leq 1 \quad (i)$$
$$r_{R_{o1}} - b_{o1} \leq 0 \qquad r_{I_{o1}} - b_{o1} = 0 \quad (ii)$$

Solution: $cost = -19$, $r_{R_B} = 0$, $r_{R_{o1}} = 1$, $r_{I_{o1}} = 1$, $b_B = 0$, $b_{o1} = 1$

Fig. 3. Node x: First ILP iteration

RMP 1b

Min
$$x_{R_{o1}I_{o1}} + 10h_{R_B} + 10h_{R_{o1}} + 10h_{I_{o1}}$$

Subject to: $$h_{R_B} \geq 1$$ $$x_{R_{o1}I_{o1}} + h_{R_{o1}} \geq 1$$ $$x_{R_{o1}I_{o1}} + h_{I_{o1}} = 1$$

Solution: $cost = 11$, $x_{R_{o1}I_{o1}} = 1$, $h_{R_B} = 1$, $h_{R_{o1}}$, $h_{I_{o1}} = 0$
Duals: $\pi_{R_B} = 10$, $\pi_{R_{o1}} = 10$, $\omega_{I_{o1}} = -9$

PP 1b

Min
$$b_B + b_{o1} - 10r_{R_B} - 10r_{R_{o1}} + 9r_{I_{o1}}$$

Subject to:

$$r_{R_B} - b_B \leq 0 \qquad b_B + b_{o1} \leq 1 \quad (i)$$
$$r_{R_{o1}} - b_{o1} \leq 0 \qquad r_{I_{o1}} - b_{o1} = 0 \quad (ii)$$

Solution: $cost = -9$, $r_{R_B} = 1$, $r_{R_{o1}} = 0$, $r_{I_{o1}} = 0$, $b_B = 1$, $b_{o1} = 0$

Fig. 4. Node x: Second ILP iteration

5. As the value of r_{R_B} is 1 in (PP 1b), the variable x_{R_B} is added to (RMP 1c). $\mathcal{P}' = \{\{R_{o1}, I_{o1}\}, \{R_B\}\}$ and the cost is further reduced from 11 in (RMP 1b) to 2 in (RMP 1c).
6. All artificial variables in (RMP 1c) are zero which might indicate that we have reached a feasible solution. The reduced cost of (PP 1c) is not negative anymore which means that (RMP 1c) cannot be improved further. Therefore, AM terminates after third ILP iteration and returns the optimal solution $\sigma(x) = \{\langle\{R\}, \{o1\}, 1\rangle, \langle\{R\}, \{B\}, 1\rangle\}$.
7. The *fil*-Rule creates two new nodes x_1 and x_2 with $\mathcal{L}(x_1) \leftarrow \{o1\}$, $\mathcal{L}(x_2) \leftarrow \{B\}$, $\sharp x_1 \leftarrow 1$ and $\sharp x_2 \leftarrow 1$.
8. The e-Rule creates edges $\langle x, x_1\rangle$ and $\langle x, x_2\rangle$ with $\mathcal{L}(\langle x, x_1\rangle) \leftarrow \{R, S\}$ and $\mathcal{L}(\langle x, x_2\rangle) \leftarrow \{R, S\}$ (because $R \sqsubseteq S \in \mathcal{R}$). It also creates back edges $\langle x_1, x\rangle$ and $\langle x_2, x\rangle$ with $\mathcal{L}(\langle x_1, x\rangle) \leftarrow \{R^-, S^-\}$ and $\mathcal{L}(\langle x_2, x\rangle) \leftarrow \{R^-, S^-\}$.
9. By unfolding B in the label of x_2 and by applying the $\sqcap$-Rule we get $\mathcal{L}(x_2) = \{B, \exists R.C, \exists R^-.D, \forall S^-.\{o2\}\}$.
10. The *inverse*-Rule encodes information about existing R^--neighbour x of x_2 by adding a tuple $\langle x, \{R^-, S^-\}\rangle$ to $\mathcal{B}(x_2)$.
11. AM uses $\{\exists R.C, \exists R^-.D\}$ to start ILP. Due to lack of space we cannot provide the complete RMP and PP solution process here. Since $R \sqsubseteq S$, the universal restriction $\forall S^-.\{o2\}$ is ensured by adding the following inequalities to PP:

RMP 1c

Min

$x_{R_{o1}I_{o1}} + x_{R_B} + 10h_{R_B} + 10h_{R_{o1}} + 10h_{I_{o1}}$

Subject to:

$$x_{R_B} + h_{R_B} \geq 1$$
$$x_{R_{o1}I_{o1}} + h_{R_{o1}} \geq 1$$
$$x_{R_{o1}I_{o1}} + h_{I_{o1}} = 1$$

Solution: *cost* $= 2$, $x_{R_{o1}I_{o1}} = 1$, $x_{R_B} = 1$, $h_{R_B}, h_{R_{o1}}, h_{I_{o1}} = 0$
Duals: $\pi_{R_B} = 1, \pi_{R_{o1}} = 10, \omega_{I_{o1}} = -9$

PP 1c

Min

$$b_B + b_{o1} - 1r_{R_B} - 10r_{R_{o1}} + 9r_{I_{o1}}$$

Subject to:

$r_{R_B} - b_B \leq 0$	$b_B + b_{o1} \leq 1 \quad (i)$
$r_{R_{o1}} - b_{o1} \leq 0$	$r_{I_{o1}} - b_{o1} = 0 \quad (ii)$

Solution:
cost $= 0$, all variables are 0.

Fig. 5. Node x: Third ILP iteration

$r_{R_\top^-} - r_{S_\top^-} \leq 0$, $r_{S_\top^-} - b_{o2} \leq 0$, and for all $r_{R_q^-}$ we added an equality $r_{R_q^-} - r_{R_\top^-} \leq 0$. Therefore, whenever $r_{R_q^-} = 1$ the values of $r_{R_\top^-}, r_{S_\top^-}, b_{o2} = 1$.

12. Since $\mathcal{B}(x_2)$ contains $\langle x, \{R^-, S^-\}\rangle$, AM adds node x in solution. Therefore, AM returns the solution $\sigma(x_2) = \{\langle\{R\}, \{C\}, 1\rangle, \langle\{R^-, S^-\}, \{D, o2\}, 1, \{x\}\rangle\}$.
13. The *fil*-Rule creates only one new node x_3 with $\mathcal{L}(x_3) \leftarrow \{C\}$ and $\sharp x_3 \leftarrow 1$, and updates the label of node x with $\mathcal{L}(x) \leftarrow \{D, o2\}$.
14. The *e*-Rule creates edges $\langle x_2, x_3\rangle$ and $\langle x_3, x_2\rangle$ with $\mathcal{L}(\langle x_2, x_3\rangle) \leftarrow \{R, S\}$ and $\mathcal{L}(\langle x_3, x_2\rangle) \leftarrow \{R^-, S^-\}$.
15. By unfolding C in the label of x_3 we get $\mathcal{L}(x_3) = \{C, \exists R.E\}$. AM gives solution $\sigma(x_3) = \{\langle\{R\}, \{E\}, 1\rangle\}$. The *fil*-Rule creates node x_4 with $\mathcal{L}(x_4) \leftarrow \{E\}$ and $\sharp x_4 \leftarrow 1$. The *e*-Rule creates edges $\langle x_3, x_4\rangle$ and $\langle x_4, x_3\rangle$ with $\mathcal{L}(\langle x_3, x_4\rangle) \leftarrow \{R, S\}$ and $\mathcal{L}(\langle x_4, x_3\rangle) \leftarrow \{R^-, S^-\}$.
16. $\mathcal{L}(x_4) = \{E, \forall S^-.\{o1\}\}$ and after unfolding E the $\forall$-Rule adds $o1$ to $\mathcal{L}(x_3)$. However, $o1$ already occurs in $\mathcal{L}(x_1)$ and $x_1 \neq x_3$. Therefore, the nom_{merge}-Rule merges node x_3 into node x_1.
17. Since no more rules are applicable, the tableau algorithm terminates.

4 Performance Evaluation

We developed a prototype system called Cicada[1] that implements our calculus as proof of concept. Besides the use of ILP and branch-and-price Cicada only implements a few standard optimization techniques such as lazy unfolding [2,17], nominal absorption [17], and dependency directed backtracking [1] as well as a *ToDo list* architecture [24] to control the application of the expansion rules. Cicada might not perform well for $\mathcal{SHOI}$ ontologies that require other optimization techniques.

Therefore, we built a set of synthetic test cases to empirically evaluate Cicada. Figure 6 presents some metrics of benchmark ontologies and evaluation results. We compared Cicada with major OWL reasoners such as FaCT++ (1.6.5) [24], HermiT (1.3.8) [21], and Konclude (0.6.2) [23].

[1] System and test ontologies: https://users.encs.concordia.ca/~haarslev/Cicada

Ontology Name	**Ontology Metrics**			**Evaluation Results**			
	#Axioms	**#Concepts**	**#Ind**	**Cic**	**FaC**	**Her**	**Kon**
EU-Members	67	32	28	4.86	TO	TO	TO
CA-Provinces	32	14	11	2.85	316.4	TO	TO

n	**Ontology Metrics**			**TestOnt-Cons Evaluation Results**				**TestOnt-InCons Evaluation Results**			
	#Axioms	**#Concepts**	**#Ind**	**Cic**	**FaC**	**Her**	**Kon**	**Cic**	**FaC**	**Her**	**Kon**
40	92	43	41	3.39	TO	TO	TO	4.41	TO	TO	TO
20	53	23	21	1.21	TO	TO	TO	3.16	TO	TO	TO
10	33	13	11	0.91	TO	TO	TO	2.68	401.7	TO	TO
7	27	10	8	0.64	1.26	3.47	3.56	2.32	1.48	3.70	3.71
5	23	8	6	0.41	0.02	0.13	0.24	2.21	0.12	0.46	0.14

Fig. 6. Metrics of Benchmark Ontologies and Evaluation Results with runtime in seconds and a timeout of 1000 seconds (TO=timeout, #=Number of..., Ind=Individuals, Cic=Cicada, FaC=FaCT++, Her=HermiT, Kon=Konclude)

The first benchmark (see top part of Figure 6) uses two real-world ontologies. The ontology EU-Members (adapted from [10]) models 28 members of European Union (EU) whereas CA-Provinces models 10 provinces of Canada. We added nominals requiring 29 EU members and 11 Canadian provinces respectively. The results show that only Cicada can identify the inconsistency of EU-Members within the time limit. Moreover, Cicada is more than two orders of magnitude faster than FaCT++ in identifying the inconsistency of CA-Provinces.

The second benchmark (see bottom part of Figure 6) consists of small synthetic test ontologies that are using a variable n for representing the number of nominals. In order to test the effect of increased number of nominals we defined concept C and A as $C \sqsubseteq \exists R^{-}.A$ and $A \sqsubseteq \exists R.X_1 \sqcap, ..., \sqcap \exists R.X_n \sqcap \forall R.\{o_1, ..., o_n\}$. Nominals $o_1, ..., o_n$ and concepts $X_1, ..., X_n$ are declared as pairwise disjoint. The first set consists of consistent ontologies in which we declared C and $X_1, ..., X_{n-1}$ as pairwise disjoint. The second set consists of inconsistent ontologies in which we declared C and $X_1, ..., X_n$ as pairwise disjoint. Only Cicada can process the ontologies with more than 10 nominals within the time limit.

5 Conclusion

We presented a tableau-based algebraic calculus for handling the numerical restrictions imposed by nominals, existential and universal restrictions, and their interaction with inverse roles. These numerical restrictions are translated into linear inequalities which are then solved by using algebraic reasoning. The algebraic reasoning is based on a branch-and-price technique that either computes an optimal solution, or detects infeasibility. An empirical evaluation of our calculus showed that it performs better on ontologies having a large number of nominals, whereas other reasoners were unable to classify them within a reasonable amount of time. In future work, we will extend the technique presented here to $\mathcal{SHOIQ}$.

References

1. Baader, F.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
2. Baader, F., Hollunder, B., Nebel, B., Profitlich, H.J., Franconi, E.: Am empirical analysis of optimization techniques for terminological representation systems. Applied Intelligence **4**(2), 109–132 (1994)
3. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Operations research **46**(3), 316–329 (1998)
4. Chvatal, V.: Linear programming. Macmillan (1983)
5. CPLEX optimizer, https://www.ibm.com/analytics/cplex-optimizer
6. Cucala, D.T., Cuenca Grau, B., Horrocks, I.: Consequence-based reasoning for description logics with disjunction, inverse roles, and nominals. In: Proceedings of the 30th International Workshop on Description Logics (July 2017)
7. Cucala, D.T., Cuenca Grau, B., Horrocks, I.: Consequence-based reasoning for description logics with disjunction, inverse roles, number restrictions, and nominals. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 1970–1976 (2018)
8. Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programs. Operations research **8**(1), 101–111 (1960)
9. Desrosiers, J., Soumis, F., Desrochers, M.: Routing with time windows by column generation. Networks **14**(4), 545–565 (1984)
10. Faddoul, J., Haarslev, V.: Algebraic tableau reasoning for the description logic SHOQ. Journal of Applied Logic **8**(4), 334–355 (2010)
11. Faddoul, J., Haarslev, V.: Optimizing algebraic tableau reasoning for SHOQ: First experimental results. In: Proceedings of the 23rd International Workshop on Description Logics (DL'10). pp. 161–171 (2010)
12. Farid, H., Haarslev, V.: Handling nominals and inverse roles using algebraic reasoning (2018), extended version of the paper published in Proceedings of the International Workshop on Description Logics (DL'18). https://arxiv.org/abs/1810.00916
13. Farsiniamarj, N., Haarslev, V.: Practical reasoning with qualified number restrictions: A hybrid Abox calculus for the description logic SHQ. AI Communications **23**(2-3), 205–240 (2010)
14. Haarslev, V., Möller, R.: Racer system description. In: Proceedings of the International Joint Conference on Automated Reasoning. pp. 701–705. Springer (2001)
15. Haarslev, V., Timmann, M., Möller, R.: Combining tableaux and algebraic methods for reasoning with qualified number restrictions. In: Proceedings of the International Workshop on Description Logics (DL'01) (2001)
16. Hladik, J.: A tableau system for the description logic SHIO. In: Proceedings of the Doctoral Programme of IJCAR. vol. 106. Citeseer (2004)
17. Horrocks, I.: Using an expressive description logic: FaCT or fiction? In: Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98). vol. 98, pp. 636–645 (1998)
18. 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)
19. Ohlbach, H., Köhler, J.: Modal logics, description logics and arithmetic reasoning. Artificial Intelligence **109**(1-2), 1–31 (1999)
20. Roosta Pour, L., Haarslev, V.: Algebraic reasoning for SHIQ. In: Proceedings of the International Workshop on Description Logics (DL'12). pp. 530–540 (2012)

21. Shearer, R., Motik, B., Horrocks, I.: HermiT: A highly-efficient OWL reasoner. In: Proceedings of the OWL: Experiences and Directions (OWLED). vol. 432, p. 91 (2008)
22. Steigmiller, A., Glimm, B., Liebig, T.: Coupling tableau algorithms for expressive description logics with completion-based saturation procedures. In: Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR'14). pp. 449–463. Springer (2014)
23. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: system description. Web Semantics: Science, Services and Agents on the World Wide Web **27**, 78–85 (2014)
24. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proceedings of the International Joint Conference on Automated Reasoning. pp. 292–297. Springer (2006)
25. Vlasenko, J., Daryalal, M., Haarslev, V., Jaumard, B.: A saturation-based algebraic reasoner for ELQ. In: Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning (PAAR 2016). pp. 110–124. CEUR (2016)
26. Vlasenko, J., Haarslev, V., Jaumard, B.: Pushing the boundaries of reasoning about qualified cardinality restrictions. In: Proceedings of the International Symposium on Frontiers of Combining Systems. pp. 95–112. Springer (2017)
27. Zolfaghar Karahroodi, N., Haarslev, V.: A consequence-based algebraic calculus for SHOQ. In: Proceedings of the International Workshop on Description Logics (DL'17) (2017)