Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
KR and Constraints

Expressive Description Logics with Rich Yet Affordable Numeric Constraints

Federica Di Stefano , Sanja Lukumbuzya , Magdalena Ortiz , Mantas Šimkus
Institute of Logic and Computation, TU Wien
{sanja.lukumbuzya|federica.stefano|magdalena.ortiz|mantas.simkus}@tuwien.ac.at

Abstract

Description Logics (DLs) excel at representing structured knowledge in several application domains, but fall very short when it comes to reasoning about their numeric aspects. We consider the expressive DL $\mathcal{ALCHOIQ}$ with closed predicates and extend it with features ranging over user-specified finite numeric intervals, feature assertions, and local additive constraints on feature values. We illustrate the power of this language for describing problems that involve ontological and numeric reasoning and study reasoning problems that go beyond satisfiability, such as finding models that minimize some costs. We show that these additional numeric modeling and reasoning capabilities can be accommodated by extending a standard reasoning technique for $\mathcal{ALCHOIQ}$ using linear inequalities, and the extension does not necessarily increase the worst-case computational cost.

1 Introduction

Description Logics (DLs) have proven to be excellent formalisms for describing different domains and reasoning about them. In DLs, domain knowledge is modeled using *concepts*, which represent sets of objects of the same class, and *roles*, which represent relations between pairs of objects. Different DLs provide different constructs to build complex *concept descriptions* in terms of simpler concepts and roles, and a detailed understanding of the tradeoff between expressiveness and complexity supports a problem-dependent choice of an adequate language. For example, in the basic $\mathcal{ALC}$ we can use the concept names $\mathsf{Employee}$ and $\mathsf{Project}$, and the role name $\mathsf{assignedTo}$, to describe as $\mathsf{Employee} \sqcap \neg(\exists\, \mathsf{assignedTo}.\mathsf{Project})$ the employees that are not assigned to a project. These expressions can be used in *concept inclusions* in a knowledge base to assert, for example, that all employees who are not in the administrative or executive category must be assigned to some project.

$$\mathsf{Employee} \sqcap \neg(\mathsf{Admin} \sqcup \mathsf{Exec}) \sqsubseteq \exists\, \mathsf{assignedTo}.\mathsf{Project}$$

The DL community has studied some very rich languages with convenient and succinct constructs for describing sophisticated domains. For example, $\mathcal{ALCHOIQ}$ extends the basic DL $\mathcal{ALC}$ with *inverse roles*, *number restrictions* and *nominals*, which we can use to say that all projects except p_1 must have at least three employees assigned.

$$\mathsf{Project} \sqcap \neg\{p_1\} \sqsubseteq\ \geq 3\, \mathsf{assignedTo}^{-}.\mathsf{Employee}$$

The DL we consider in this paper goes further and allows us to say that certain concepts and roles are *closed*, which means that we are given their extension explicitly, and no further objects can be added. E.g., if the concept $\mathsf{Project}$ is closed and the projects p_1, p_2 and p_3 are listed in the knowledge base, then we know that these are the project employees must be assigned to one of the three, and not to some other unlisted project. $\mathcal{ALCHOIQ}$ with closed predicates is among the most flexible knowledge representation languages and it has been shown decidable, and reasoning techniques and worst-case upper bounds are available.

However, a significant weakness of DLs—even of very expressive ones like $\mathcal{ALCHOIQ}$ with closed predicates—is their very limited support for numeric constraints and quantitative reasoning. In many applications, one would like to be able to refer to integers or real numbers and basic comparisons between them. We can describe complex requirements on the assignment of employees to projects, as we have illustrated, but we cannot reason about the number of hours that employees work for a project, or verify that the sum of the monthly salary of all employees assigned to a project does not exceed the corresponding budget. Standard DLs cannot express this type of quantitative constraints, and adding them is far from easy. DLs adopt the open-world semantics of classical predicate logic, and their support for existential quantification enables DLs to describe (often infinite) sets of unnamed objects. In the presence of these open domains, even simple numeric reasoning easily becomes computationally very costly, or even undecidable.

Overcoming this limitation is a long-standing challenge that has received significant attention since the early days of DL research. The most prominent line of work here is *concrete domains* (Baader and Hanschke 1991). Additionally to concepts and roles, we use concrete *features* such as age, salary, size, etc. to relate objects to values from a domain, like the reals or the integers, for example. A fixed set of predicates over these domains, such as addition and comparisons, can then be used to incorporate numeric constraints into concept descriptions. DLs with concrete domains are very powerful, but this comes at a high computational cost. Strong restrictions must be imposed on the concrete domains to preserve decidability (see (Borgwardt, De Bortoli, and Koopmann 2024; Lutz 2002) and their references). Most works require the so-called ω-admissibility,

which in a nutshell guarantees that the infinite systems of constraints that may arise when reasoning about infinite models can be effectively decided. We have tight bounds for standard reasoning tasks like concept satisfiability and instance checking for expressive DLs with ω-admissible concrete domains (Borgwardt, De Bortoli, and Koopmann 2024; Lutz 2002)—and even a few isolated results for non-ω-admissible ones (Labai, Ortiz, and Šimkus 2020; Demri and Quaas 2023)—but those works focus on worst-case bounds and generic restrictions that preserve decidability many domains, rather than on their practical usability. In contrast, efforts to support powerful and useful domains by restricting the interaction with the logics, and practicable algorithms for them, have been extremely limited (Haarslev, Möller, and Wessel 2001; Pan and Horrocks 2002).

When DLs are extended with numeric features, many novel and natural reasoning problems arise. For example, if a feature corresponds to a cost, it is natural to minimize it: a company may not only be interested in staying within a given budget when fulfilling certain requirements, but may naturally want to minimize their cost. Given the open domains of DLs, the cost associated to a specific feature may not always be finite, so it is natural to ask whether the KB can be satisfied while guaranteeing that certain cost remains bounded, or below a certain value. Popular reasoning problems such as concept satisfiability, instance checking and query answering can all be defined in terms of optimal models. Additionally, we can ask what values certain features might take when other features are optimized. Despite their evident potential, to our knowledge, this kind of numeric reasoning service has not yet been developed for DLs.

In this paper we take a pragmatic approach and enhance very expressive DLs with simple yet useful numeric reasoning capabilities, and propose a toolbox of reasoning services that seamlessly integrate ontological and numeric reasoning. We use features to assign numeric values to domain objects, and the concept descriptions in our knowledge bases may comprise constraints on the sums of the feature values in the neighborhood of objects. As usual in DLs, domains may be arbitrarily large, but we require a finite bound on the range of possible feature values of each object. In this way, decidability is not compromised, even in very expressive DLs like $\mathcal{ALCHOIQ}$ and its extension with closed predicates. Reasoning in our numeric extension can be seamlessly achieved with the standard reasoning algorithm for $\mathcal{ALCHOIQ}$ with closed predicates by reduction to a system of linear inequalities. We identify some conditions on the numeric ranges that guarantee that the worst-case complexity of reasoning is not higher than for plain $\mathcal{ALCHOIQ}$. A highlight of our approach is its natural support for many novel reasoning services, thus revealing a new tool for flexible quantitative inference in the presence of rich ontological knowledge.

2 Preliminaries

Description Logics. We refer to (Baader et al. 2017) for preliminaries on Description Logics. Let N_C, N_R, and N_I be countably infinite, mutually disjoint sets of *concept names*, *role names*, and *individuals*, respectively. An *assertion* (or, *fact*) is an expression of the form $r(a,b)$, $\neg r(a,b)$, $A(b)$ or $\neg A(b)$, where $r \in N_R$, $A \in N_C$, and $a, b \in N_I$. An *ABox* $\mathcal{A}$ is a finite set of assertions. We next recall the DL $\mathcal{ALCHOIQ}$. Let N_R^+ denote the set of *roles* defined as follows: $N_R^+ = \{p, p^- \mid p \in N_R\}$. With a slight abuse of notation, we write r^- to denote p^- if $r = p$, and p if $r = p^-$, for $p \in N_R$. The role r^- is the *inverse* of r. If $o \in N_I$, then $\{o\}$ is called a *nominal*, and the set N_C^+ of *basic concepts* is defined as $N_C^+ = N_C \cup \{\{o\} \mid o \in N_I\} \cup \{\top, \bot\}$. *(Complex) concepts* are defined according to the following syntax:

$$C := A \mid C \sqcap C \mid C \sqcup C \mid \neg C \mid \{o\} \mid \exists r.C \mid \forall r.C \mid \circ\, n\, r.C,$$

where $A \in N_C$, $o \in N_I$, $r \in N_R^+$, and $\circ \in \{\leq, \geq\}$. *Axioms* take the forms $C \sqsubseteq D$ (*concept inclusions*) and $r \sqsubseteq p$ (*role inclusions*), where C, D are concepts and r, p are roles. A *TBox* is a finite set of axioms. A *knowledge base (KB)* is a tuple $\mathcal{K} = (\mathcal{T}, \Sigma, \mathcal{A})$, where $\mathcal{T}$ is a TBox, $\Sigma \subseteq N_C \cup N_R$ is a set of *closed predicates* and $\mathcal{A}$ is an ABox.

We use $N_C(\mathcal{X}), N_C^+(\mathcal{X}), N_R(\mathcal{X}), N_R^+(\mathcal{X}), N_I(\mathcal{X})$ to, respectively, denote the concept names, basic concepts, role names, roles, and individuals that occur in $\mathcal{X}$, where $\mathcal{X}$ is an ABox, a TBox, or a KB. The semantics of TBoxes and ABoxes as given above is defined in the standard way via first-order *interpretations* of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. Given an interpretation $\mathcal{I}$, $d \in \Delta^{\mathcal{I}}$ and $r \in N_R^+$, an element $e \in \Delta^{\mathcal{I}}$ is an *r-neighbor* of d if $(d,e) \in r^{\mathcal{I}}$. The *neighborhood* of an element d is the set of all its r-neighbors, with $r \in N_R^+$. We say that $\mathcal{I}$ satisfies a KB $\mathcal{K} = (\mathcal{T}, \Sigma, \mathcal{A})$, in symbols $\mathcal{I} \vDash \mathcal{K}$, if $\mathcal{I}$ satisfies $\mathcal{T}$ and $\mathcal{A}$ and (i) for each $A \in \Sigma$, $A^{\mathcal{I}} = \{a \mid A(a) \in \mathcal{A}, A \in \Sigma \cap N_C\}$, and (ii) for each $r \in \Sigma$, $r^{\mathcal{I}} = \{(a,b) \mid r(a,b) \in \mathcal{A}, r \in \Sigma \cap N_R\}$. See (Lukumbuzya, Ortiz, and Šimkus 2024) for more details. Finally, we make the *standard name assumption* (SNA).

Enriched Integer Linear Programs. An *integer linear inequality* is an expression of the form $a_1x_1 + \cdots + a_nx_n \leq b_1y_1 + \ldots b_my_m$, where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are non-negative integer coefficients and $x_1, \ldots, x_n, y_1, \ldots, y_m$ are variables. A system of integer linear inequalities is a tuple $(V, \mathcal{E})$, where V is a set of variables and $\mathcal{E}$ is a set of integer linear inequalities in the variables from V. An *integer linear program (ILP)* Π is a tuple $(V, \mathcal{E}, \min c_1x_1 + \cdots + c_nx_n)$, where $(V, \mathcal{E})$ is a system of integer linear inequalities, $c_1, \ldots, c_n$ are integer coefficients and $x_1, \ldots, x_n \in V$. The expression $\min c_1x_1 + \cdots + c_nx_n$ is the *objective function of* Π. Given a numeric domain $\mathcal{D}$ and a system of inequalities $\mathcal{S} = (V, \mathcal{E})$, a function $S : V \to \mathcal{D}$ is a *solution* to $\mathcal{S}$ over $\mathcal{D}$ if, for all variables $x \in V$, substituting $S(x)$ for x satisfies the inequalities of $\mathcal{E}$. For the ILP $\Pi = (V, \mathcal{E}, \min c_1x_1 + \cdots + c_nx_n)$, the value $c_1S(x_1) + \ldots c_nS(x_n)$ is the *objective value of* S. We say that S is an *optimal solution* of Π if there is no other solution S' such that the objective value of S' is smaller than the objective value of S. The objective value of S is then called the *optimal value of* Π.

An *enriched system (of integer linear inequalities)* is a tuple $(V, \mathcal{E}, I)$, where $(V, \mathcal{E})$ is a system of integer linear inequalities and I is a set of implications of the form $q_1 \Rightarrow q_2$ with q_1 and q_2 linear inequalities whose variables are in V. *Enriched integer linear programs (EILPs)* are defined similarly to ILPs but use enriched systems. All other notions are

defined analogously.

For our purposes, we will be interested in solving enriched systems over $\mathbb{N}^* = \mathbb{N} \cup \{\aleph_0\}$, where $\aleph_0$ represents infinity. We extend the total order $\leq$ over $\mathbb{N}$ to total order over $\mathbb{N}^*$ by setting $n \leq \aleph_0$, for every $n \in \mathbb{N}$ and $\aleph_0 \leq \aleph_0$; arithmetic operations are extended to $\mathbb{N}^*$ as usual.

Proposition 1. *(Gogacz et al. 2019; Lukumbuzya, Ortiz, and Šimkus 2024) Given a finite enriched system $\mathcal{S} = (V, \mathcal{E}, I)$ in which all coefficients are in $\{0, \pm 1, \ldots, \pm a\}$, the following hold:*

1. *If $(V, \mathcal{E}, I)$ has a solution over $\mathbb{N}$, then it also has a solution over $\mathbb{N}$ where all values are bounded by $(|V| + |\mathcal{E}| + |I|) \cdot ((|\mathcal{E}| + |I|) \cdot a)^{2(|\mathcal{E}|+|I|)+1}$.*
2. *If $(V, \mathcal{E}, I)$ has a solution over $\mathbb{N}^*$, then it also has a solution over $\mathbb{N}^*$ where all finite values are bounded by $(|V| + |\mathcal{E}| + |I|) \cdot ((|\mathcal{E}| + |I|) \cdot a)^{2(|\mathcal{E}|+|I|)+1}$.*
3. *Deciding whether $\mathcal{S}$ has a solution over $\mathbb{N}^*$ (resp. $\mathbb{N}$) is in NP.*

The following result follows from (Lukumbuzya and Šimkus 2021) and (Papadimitriou 1981)

Proposition 2. *Consider a finite EILP $\Pi = (V, \mathcal{E}, I, \min c_1 x_1 + \cdots + c_n x_n)$ in which all coefficients are in $\{0, \pm 1, \ldots, \pm a\}$. Deciding whether Π has a finite optimal value is possible in coNP. If Π has a finite optimal value b, then $|b| \leq \sum_{i=1}^{n} |c_i| \cdot |V|^2((|\mathcal{E}| + |I|)a)^{2(|\mathcal{E}|+|I|)+3}$.*

3 Adding Numeric Features

In this section we present our formalism that enriches standard DLs (with closed predicates) with numeric features. We begin with a short motivating scenario, which serves as a running example for this section.

Example 1. *Consider a scenario in which someone wants to set up a new company. According to the financial plan, they have a yearly expense budget to comply with. For simplicity, we assume that they need to hire three different types of employees: at least one executive officer, one administrative employee, and one regular employee; other TBox axioms and ABox assertions could describe the projects planned for the first year and the corresponding personnel requirements. Each employee category comes with a predefined range for the salary, which can be negotiated and adapted to the level of expertise and is a monthly expense for the company. The employees will be given office space and the necessary equipment, thus the company has to take care of monthly rent and one-time equipment costs.*

$$\{comp\} \sqsubseteq \exists \mathsf{hire}.\mathsf{Exec} \sqcap \exists \mathsf{hire}.\mathsf{Adm} \sqcap \exists \mathsf{hire}.\mathsf{RegEmpl}$$
$$\mathsf{Empl} \sqsubseteq \exists \mathsf{hasOff}.\mathsf{Office} \sqcap \exists \mathsf{hasEq}.\mathsf{Laptop} \sqcap \exists \mathsf{hasEq}.\mathsf{Desk}$$
$$\mathsf{Exec} \sqcup \mathsf{Adm} \sqcup \mathsf{RegEmpl} \sqsubseteq \mathsf{Empl}$$
$$\mathsf{Laptop} \sqcup \mathsf{Desk} \sqsubseteq \mathsf{Equipment}$$

We omit the axioms stating the disjointness between employees, equipment, offices, etc. Further knowledge about the company and its requirements may be expressed in the TBox and ABox, and a DL reasoner may be used to find possible solutions to the requirements. But we do not have a direct way of modeling individual salaries, costs of equipment and office space, maximum occupancies of offices, and no means of ensuring that the company's budget is respected.

We want to enrich DLs so that, in scenarios like the above, we can describe and reason about numeric aspects like costs. For example, we want to be able to assign possible salaries to employees within the described ranges, and control how some feature values for the objects compare to aggregated feature values of its neighbors. For example, we would like the sum of yearly salaries of the employees of the company $comp$ not to exceed the pre-determined yearly budget.

Syntax. We define our DL extension to standard DLs, which is expressive enough to model our example and, as we discuss below, it remains decidable. We introduce *features*, which assign numeric values to domain elements, and a new form of concept expressions that restricts the values of the sums of features in the neighborhood of an object.

Definition 1. *Let $\mathcal{L}$ be a DL (possibly with closed predicates), and let N_F be a countably infinite set of* features*, disjoint from N_C, N_R and N_I. A* feature annotation *α is a partial function from $N_C^+ \times N_F$ to finite discrete subsets of the non-negative rationals $\mathbb{Q}^+ \cup \{0\}$. We call the set $D = \alpha(B, f)$ the* domain *of feature f for B.*

A neighborhood restriction *takes the form*

$$w_0 + w_1 \sum f_1[r_1.C_1] + \cdots + w_n \sum f_n[r_n.C_n] \circ wf,$$

where $\circ \in \{<, \leq, \geq, >\}$, $r_1, \ldots, r_n$ are roles, $C_1, \ldots, C_n$ are concepts, $f, f_1, \ldots, f_n$ are features, and $w, w_1, \ldots, w_n$ are non-negative rational numbers. Concepts in $\mathcal{L}^{\mathrm{NF}}$ are defined as for $\mathcal{L}$, but allowing also neighborhood restrictions. An annotated KB in $\mathcal{L}^{\mathrm{NF}}$ is defined as a 4-tuple $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, where $(\mathcal{T}, \Sigma, \mathcal{A})$ is an $\mathcal{L}^{\mathrm{NF}}$ KB with closed predicates and α is a feature annotation function. In the following, we refer to an annotated KB simply as KB.

The idea behind our formalism is as follows. The feature annotation function α determines the set D of values that objects participating in a basic concept A can take as the value of feature f. In our case, we restrict our attention to sets D that consist of finitely many, non-negative rational values. We may specify D by a minimum and a maximum value together with a fixed 'step size', e.g., $D = [0, 1, +0.25] = \{0, 0.25, 0.5, 0.75, 1\}$. In our example, $\alpha(\mathsf{Exec}, salary) = [3.5, 5.5, +0.1]$ indicates that executive employees can have their $salary$-value anywhere between 3.5k and 5.5k, with increases of 100€. Note that α is a partial function, meaning that $\alpha(A, f)$ may not be defined. For instance, we may not want to prescribe values for $\alpha(\mathsf{Office}, salary)$, since offices have no salary. We will see below that features will be interpreted as partial functions, and domain objects whose f is not prescribed will have an undefined f value. The intuitive meaning of α is that, if $\alpha(A, f)$ is defined, then, in every model of the KB, an object in A *must* pick exactly one value from $\alpha(A, f)$ as its f-value. An expression $\Sigma f[r.C]$ intuitively stands for sum of the f-values of all r-neighbors satisfying the concept C.

Example 2. *In our example scenario, we can use the extended syntax to assign salaries, costs, and budget constraints for the company. We let $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, where*

$\mathcal{T}$, Σ, and $\mathcal{A}$ are as described in the KB of Example 1, and additionally we have a feature annotation α with

$$\begin{aligned}\alpha(\mathsf{Exec}, \mathit{salary}) &= [3.5, 5.5, +0.1],\\ \alpha(\mathsf{Adm}, \mathit{salary}) &= [2.2, 3.1, +0.1],\\ \alpha(\mathsf{RegEmpl}, \mathit{salary}) &= [3.1, 4.5, +0.1],\\ \alpha(\mathsf{Office}, \mathit{capacity}) &= [0, 6, +1],\\ \alpha(\mathsf{Office}, \mathit{rent}) &= [0.3, 0.6, +0.05],\\ \alpha(\mathsf{Laptop}, \mathit{cost}) &= [0.5, 2, +0.05],\\ \alpha(\mathsf{Desk}, \mathit{cost}) &= \{0.2\},\\ \alpha(\{\mathit{comp}\}, \mathit{budget}) &= [0, 500, +20]\end{aligned}$$

and $\alpha(A, f)$ is undefined otherwise. With this annotation, we associate each type of employee with a feature salary*, for each office we state (with two features) the* capacity *and the* rent *cost. Finally, with the last annotation, we determine the global* budget *of the company. We now ensure that our company's budgets and maximum occupancy restrictions are respected. For this, we require that 12 times the sum of the monthly salaries of all its employees, plus 12 times the sum of monthly rent for all the offices the company rents plus the sum of all one-time equipment expenses is not higher than the global budget of the company. We can formalize such a requirement with the concept C defined as follows:*

$$12\cdot \sum \mathit{salary}[\mathsf{hire.Empl}] + 12 \cdot \sum \mathit{rent}[\mathsf{hasOff.Office}] + \sum \mathit{cost}[\mathsf{hasEq.Equipment}] \leq \mathit{budget}$$

To ensure that the company comp satisfies C, we add to $\mathcal{T}$ the inclusion $\{\mathit{comp}\} \sqsubseteq C$. To ensure that the maximum occupancy of offices is respected, we consider a new feature pers that 'counts' the number of employees sitting in one office. We then obtain $\mathcal{K}' = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$ by letting $\alpha(\mathsf{Empl}, \mathit{pers}) = \{1\}$ and adding to $\mathcal{T}$ the inclusion

$$\mathsf{Office} \sqsubseteq \sum \mathit{pers}[\mathsf{hasOff}^-.\mathsf{Employee}] \leq \mathit{capacity}.$$

Semantics. To define the semantics of our language extension, we first need to extend the usual notion of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ to interpret the features. To this end, the interpretation function $\cdot^{\mathcal{I}}$ assigns to every feature $f \in N_F$ a partial function $f^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to non-negative rational values.

The semantics of neighborhood restrictions $w_0 + w_1 \sum f_1[r_1.C_1] + \cdots + w_n \sum f_n[r_n.C_n] \circ wf$ is now straightforward: at an element e, we compare the feature value of f multiplied by the weight w with the weighted sum of the feature values of the relevant neighbors of e. In particular, if d is an r_i-neighbor of e that participates in C_i, then the value $f_i^{\mathcal{I}}(e)$ participates in this sum. Formally:

$$\begin{aligned}&(w_0 + w_1 \sum f_1[r_1.C_1] + \cdots + w_n \sum f_n[r_n.C_n] \circ wf)^{\mathcal{I}}\\ &= \{o \in \Delta^{\mathcal{I}} : \big(w_0 + \sum_{\substack{(o,o')\in r_i^{\mathcal{I}}, o'\in C_i^{\mathcal{I}},\\ f_i^{\mathcal{I}}(o') \text{ def.}, 1\leq i\leq n}} w_i f_i^{\mathcal{I}}(o')\big) \circ wf^{\mathcal{I}}(o)\}\end{aligned}$$

Satisfaction of axioms in $\mathcal{I}$ involving the new concept constructor is defined as usual.

Definition 2. *Given a KB $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, an interpretation $\mathcal{I}$ is* $\mathcal{K}$-suitable*, if for each $e \in \Delta^{\mathcal{I}}$, each feature $f \in N_F$, and each basic concept $A \in N_C^+(\mathcal{K})$ with $e \in A^{\mathcal{I}}$ and $\alpha(A, f)$ defined, we have that $f^{\mathcal{I}}(e) \in \alpha(A, f)$. We say that $\mathcal{I}$* satisfies *$\mathcal{K}$, and call $\mathcal{I}$ a* model of *$\mathcal{K}$, if (i) $\mathcal{I}$ is $\mathcal{K}$-suitable and (ii) $\mathcal{I}$ satisfies $(\mathcal{T}, \Sigma, \mathcal{A})$.*

Example 3. *Assume a company with one executive, Ann, two regular employees, Alice and Bob, and one admin, Karl, who occupy a total of 3 offices, one per role; each employee has their own desk and laptop. The company can be represented as a model $\mathcal{I}$ of the KB $\mathcal{K}$ described in Example 1, and this model $\mathcal{I}$ can be extended to a model of the $\mathcal{K}'$ from Example 2, which has feature annotations and additional axioms, by setting the interpretation of features as follows:*

$$\begin{aligned}&\mathit{budget}^{\mathcal{I}}(\{\mathit{comp}\}) = 220 && \mathit{cost}^{\mathcal{I}}(\mathit{lap}_i) = 1\\ &\mathit{cost}^{\mathcal{I}}(\mathit{desk}_i) = 0.2 && \mathit{rent}^{\mathcal{I}}(\mathit{off}_j) = 0.5\\ &\mathit{capacity}^{\mathcal{I}}(\mathit{off}_j) = 3 && \mathit{pers}^{\mathcal{I}}(p) = 1\\ &\mathit{salary}^{\mathcal{I}}(\mathit{Ann}) = 4 && \mathit{salary}^{\mathcal{I}}(\mathit{Alice}) = 3.4\\ &\mathit{salary}^{\mathcal{I}}(\mathit{Bob}) = 3.1 && \mathit{salary}^{\mathcal{I}}(\mathit{Karl}) = 2.7\end{aligned}$$

with $p \in \{\mathit{Ann}, \mathit{Alice}, \mathit{Bob}, \mathit{Karl}\}$, $1 \leq i \leq 4$ (total number of employees) and $1 \leq i \leq 3$ (total number of offices). One can see that the budget of the company is satisfied, i.e. $\mathit{comp}^{\mathcal{I}} \in C^{\mathcal{I}}$, and that the office's capacity is not exceeded. For each $f \in N_F(\mathcal{K}')$ and $e \in \Delta^{\mathcal{I}}$, $f^{\mathcal{I}}(e) \in \alpha(A, f)$, we have that $\mathcal{I}$ is $\mathcal{K}'$-suitable. Hence $\mathcal{I}$ is a model of $\mathcal{K}'$.

Simulating $\mathcal{Q}$. We note briefly that concepts of the form $\geq n\, r.C$ and $\leq n\, r.C$, as allowed in the syntax of concept expressions, can be simulated using neighborhood restrictions and fresh features. Given $\circ n\, r.C$ with $\circ \in \{\geq, \leq\}$, let $f_{r.C}$ and $f_{=n}$ be fresh features, and let $\alpha(C, f_{\circ nr.C}) = \{1\}, \alpha(\top, f_{=n}) = \{n\}$. Then the concept $\sum f_{r.C}[r.C] \circ f_{=n}$ consists of all domain elements that have $\circ n$ r-neighbors that are C.

3.1 Decidability and Complexity

We now look at the very expressive $\mathcal{ALCHOIQ}$ with closed predicates and show that it can be enriched with numeric features and constraints at no additional cost.

For our complexity results, we define the size of a knowledge base $\mathcal{K}$ as the number of bits required to encode $\mathcal{K}$, under the assumption that the numerical values occurring in $\mathcal{K}$ are coded in unary. That is, we assume $p + q + 1$ bits are used to encode the rational value $\frac{p}{q}$.

Theorem 1. *Deciding satisfiability of $\mathcal{ALCHOIQ}^{\mathrm{NF}}$ KBs with closed predicates is* NEXPTIME*-complete.*

The rest of this section serves as a proof sketch for Theorem 1. We first recall the procedure from (Lukumbuzya, Ortiz, and Šimkus 2024), which reduces the satisfiability problem for $\mathcal{ALCHOIQ}$ KBs with closed predicates to the feasibility problem for integer programs. The main idea behind the reduction is as follows. Let $\mathcal{K}$ be some KB. The domain elements in potential models of $\mathcal{K}$ are described using structures called *star-types* or *tiles* in the literature. More

precisely, a tile τ describing a domain element d in some potential model of $\mathcal{K}$ is defined by (i) its *unary type* consisting of all the concepts names and nominals that d participates in and (ii) the relevant neighborhood of d. Each relevant neighbor e of d is further described by a role type that specifies the type of connection this neighbor has to d and its unary type. We call d an instance of τ. Each tile satisfies a set of conditions that ensures local consistency of its instances with the axioms and assertions in $\mathcal{K}$. Furthermore, an enriched system of integer linear inequalities can be extracted from $\mathcal{K}$ that describes how the number of instances of different tiles relate to each other in models of $\mathcal{K}$. It is then shown that deciding whether $\mathcal{K}$ has a model can be reduced to deciding whether this system has a solution over the set $\mathbb{N}^*$ of natural numbers extended with the value that represents infinity. Since the obtained system of inequalities is exponentially-sized in the size of $\mathcal{K}$, the NEXPTIME upper bound follows.

We explain how this procedure can be modified to accommodate our numeric features and constraints. Assume an arbitrary enumeration $f_1, \ldots, f_{N_F(\mathcal{K})}$We introduce the notion of the *feature type* $F = (v_1, \ldots, v_{|N_F(\mathcal{K})|}) \in (\mathbb{Q}^+ \cup \{0, nil\})^{|N_F(\mathcal{K})|}$, where the i-th position in F stores the value of the feature f_i of the domain element with the feature type F, for all $i, 1 \leq i \leq |N_F(\mathcal{K})|$. Moreover, we use nil for the case when the feature value is not defined.

We next modify the notion of the tile as follows. A domain element d is now described through its (i) unary type, (ii) its *feature type*, and (iii) its relevant neighborhood, where each neighbor e is described through d's connections to e, and the unary and feature type of e. We then need to add further conditions to the tile to ensure compatibility of d's feature vector with the feature annotation of the KB, as well as to ensure that all axioms that involve feature values are respected. We summarize this below.

Normal form. It will be convenient to assume in what follows that KBs are in a certain restricted form.

Definition 3. *A KB $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$ is in* normal form *if the following are satisfied.*

- *Each axiom in $\mathcal{T}$ is one of the following forms:*

(N1) $B_1 \sqcap \cdots \sqcap B_{k-1} \sqsubseteq B_k \sqcup \cdots \sqcup B_m$,

(N2) with $\circ \in \{<, \leq, \geq, >\}$,

$$B \sqsubseteq w_0 + w_1 \sum f_1[r_1.B_1] + \ldots + w_n \sum f_n[r_n.B_n] \circ w\, f,$$

(N3) $B_1 \sqsubseteq \forall r.B_2$, *or*

(N4) $r \sqsubseteq s$.

Here $\{B, B_1, \ldots, B_m\} \subseteq N_C^+$, $\{s, r, r_1, \ldots, r_n\} \subseteq N_R^+$, $\{w, w_0, \ldots, w_n\} \subseteq \mathbb{Q}^+ \cup \{0\}$, $\{f, f_1, \ldots, f_n\} \subseteq N_F$.

- $\mathcal{K}$ *is* closed under role inclusions*, i.e., it satisfies:*
  - $p \sqsubseteq p \in \mathcal{T}$*, for all* $p \in N_R(\mathcal{K})$,
  - $\{p \sqsubseteq r, r \sqsubseteq s\} \subseteq \mathcal{T}$ *implies* $p \sqsubseteq s \in \mathcal{T}$*, and*
  - $p \sqsubseteq r \in \mathcal{T}$*, then* $r^- \sqsubseteq p^- \in \mathcal{T}$.

We may call axioms of the form (N2) *neighborhood constraints*; moreover, it is an *at-most* neighborhood constraint if $\circ \in \{<, \leq\}$, and an *at-least* one otherwise.

We can now explain the additional local consistency condition that we place on our 'good' tiles in order to ensure that they describe objects that satisfy the neighborhood constraints. Consider a neighborhood constraint β of the form $B \sqsubseteq w_0 + w_1 \sum f_1[r_1.B_1] + \cdots + w_n \sum f_n[r_n.B_n] \circ wf$ with $\circ \in \{<, \leq, >, \geq\}$ and let $\mathcal{I}$ be some interpretation. We say that a domain element $e \in \Delta^{\mathcal{I}}$ is *relevant for* β if there is some i, $1 \leq i \leq n$, with $w_i \neq 0$ such that $e \in B_i^{\mathcal{I}}$, $(d, e) \in r_i^{\mathcal{I}}$ and $f_i^{\mathcal{I}}(e)$ is defined and non-zero.

A natural, but naive way to add a condition that ensures that an instance of some tile τ respects the counting β is to simply ensure that (i) all tiles with the concept A in their unary type store information about *all relevant neighbors for* β and (ii) that the weighted sum of the stored neighbors' features relates correctly to the value of the feature f in the tile's feature type. This approach works for at-most axioms, as the fact that we consider only non-negative feature values and weights ensures that only polynomially many domain elements are actually relevant for these axioms.

However, in the case of at-least axioms, we have an additional challenge, since the same bound does not apply in general. In order to ensure that we can restrict ourselves to a bounded number of neighbors that the tiles have to store we need to make one last assumption on the shape of KBs.

Definition 4. *We call $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$ a* bounded neighborhood KB *if for every at-least neighborhood constraint in $\mathcal{T}$ of the form*

$$A \sqsubseteq w_0 + w_1 \sum f_1[r_1.B_1] + \cdots + w_n \sum f_n[r_n.B_n] \circ wf$$

where $\circ \in \{>, \geq\}$, there is in $\mathcal{T}$ an at-most neighborhood constraint of the form

$$A \sqsubseteq w_0' + w_1' \sum f_1'[r_1.B_1] + \cdots + w_n' \sum f_n'[r_n.B_n] \leq wf'$$

such that $\alpha(A, f'), \alpha(B_1, f_1'), \ldots, \alpha(B_n, f_n')$ are defined and do not contain 0.

These assumptions do not restrict the expressiveness of our formalism: we show in the appendix how each KB $\mathcal{K}$ can be transformed in polynomial time into a bounded neighborhood KB $\mathcal{K}'$ in normal form. The normalization is standard. For the bounded neighborhoods, however, we must show that also for at least constraints we can infer a bound on number of elements we need to consider, and bound the corresponding neighborhood with a sufficiently large number.

Proposition 3. *Given $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, we can obtain from $\mathcal{K}$ in polynomial time a bounded neighborhood KB $\mathcal{K}' = (\mathcal{T}', \Sigma, \alpha', \mathcal{A})$ in normal form such that:*

- *every model $\mathcal{I}'$ of $\mathcal{K}'$ is also a model of $\mathcal{K}$;*
- *every model $\mathcal{I}$ of $\mathcal{K}$ can be extended into a model of $\mathcal{K}'$ by suitably interpreting the fresh predicates in $\mathcal{K}'$; and*
- *for all models $\mathcal{I}$ of $\mathcal{K}'$ and $d \in \Delta^{\mathcal{I}}$, the number of neighbors of d that are relevant for any given neighborhood constraint of $\mathcal{K}'$ is at most polynomial in the size of $\mathcal{K}$.*

The bounded neighborhoods help us adapt the reduction from the literature, allowing us to shown that KB satisfiability in our formalism reduces to solving an exponentially sized enriched system of integer linear inequalities.

Proposition 4. *Let $\mathcal{K}$ be a KB. We can obtain from $\mathcal{K}$ an enriched system $\mathcal{S}_\mathcal{K} = (V_\mathcal{K}, \mathcal{E}_\mathcal{K}, \mathcal{I}_\mathcal{K})$ of integer linear inequalities that has the following properties:*

1. *V is the set of tiles for $\mathcal{K}$,*
2. *$|V|, |\mathcal{E}|$ and $|I|$ are exponential in the size of $\mathcal{K}$,*
3. *the numerical value of the coefficients in $\mathcal{S}_\mathcal{K}$ is at most polynomial in the size of $\mathcal{K}$,*
4. *for every model $\mathcal{I}$ of $\mathcal{K}$ there exists a solution $S_\mathcal{I}$ of $\mathcal{S}_\mathcal{K}$ over $\mathbb{N}^*$ such that for each tile τ, the number of instances of τ in $\mathcal{I}$ is equal to $S_\mathcal{I}(\tau)$, and*
5. *for every solution S of $\mathcal{S}_\mathcal{K}$ over $\mathbb{N}^*$ there is a model $\mathcal{I}_S$ such that for each tile τ, the number of instances of τ in $\mathcal{I}_S$ is equal to $S(\tau)$.*

Recall that we can decide in NP whether an enriched system of integer linear inequalities has a solution over $\mathbb{N}^*$. Therefore, as a consequence of Proposition 4, we have the the result in Theorem 1 follows.

4 Reasoning with Optimal Models

Our formalism naturally supports other non-standard reasoning problems. We can easily imagine examples in which we would like to consider only those models in which the total value of certain features is optimal. We consider a brief motivating example for the basic scenario.

Example 4. *Consider the KB $\mathcal{K}'$ defined in Example 2 and the model $\mathcal{I}$ defined in Example 3. Let $\mathcal{K}''$ be the KB obtained updating $\mathcal{K}'$ with the inclusion(s)*

$$\exists \mathsf{hasOffice}^-.E \sqsubseteq \forall \mathsf{hasOffice}^-.E$$

where $E \in \{\mathsf{Adm}, \mathsf{RegEmp}, \mathsf{Exec}\}$. The company described by $\mathcal{I}$ associates each office to each type of employee, meaning that Alice and Bob share the same office, while Karl and Ann have two separate offices. Observe that each office off_j has capacity 3. Assume now that the company described by $\mathcal{I}$ expands by hiring more regular employees, administrators, etc. Even if the budget is respected, our company wants to be cost-efficient and minimize the amount it pays for rent. Intuitively speaking, the model $\mathcal{I}$ should be efficiently updated *by assigning existing offices to the newly hired people. To this end, we want to be able to specify that the feature $rent$ associated with the cost of renting an office should be* minimal *when selecting models of $\mathcal{K}''$.*

In case we want to optimize several features, there are multiple ways to define optimal models. Here we only consider the case where we want to optimize the total cost of a model. A natural example of this would be that the company prefers to keep their total expenses low, but does not care about minimizing the rent or the salaries separately.

We now proceed to formally define the notion of *optimal models* of KBs w.r.t. some given objective defined by means of a *cost function*.

Definition 5. *Given a KB $\mathcal{K}$, a* cost function for $\mathcal{K}$ *is an expression F of the form*

$$w_0 + \sum_{i=1}^{n} w_i \cdot f_i[B_i] \tag{1}$$

where $w_0, w_i \in \mathbb{Q}^+$ are weights*, $f_i \in N_F(\mathcal{K})$, and $B_i \in N_C^+(\mathcal{K})$, for all $1 \leq i \leq n$. Given a cost-function F and an interpretation $\mathcal{I}$, the* value of $\mathcal{I}$ w.r.t. F*, denoted $v_F(\mathcal{I})$, is defined as*

$$v_F(\mathcal{I}) = w_0 + \sum_{i=1}^{n} \sum_{d \in B_i^\mathcal{I} \text{ s.t. } f_i^\mathcal{I}(d) \text{ def.}} w_i f_i^\mathcal{I}(d).$$

Before formally defining *optimal models* w.r.t a cost function F, it is worth mentioning that, since our logic is expressive enough to enforce infinite models (folklore result, (Baader et al. 2017)), we may have interpretations whose value w.r.t. F is infinite. One question naturally arises: What are the intended optimal models of the KB that only has models with infinite values for a given cost function? Here, we have two options:

- All models are considered optimal models.
- There exists no optimal model. In this case, an error could be reported to the user as the problem specification is likely wrong.

We opt here for the second choice to keep our definitions in line with traditional integer programming.

Definition 6. *Let $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$ be a KB and F a cost function for $\mathcal{K}$. Given an interpretation $\mathcal{I}$, we say that $\mathcal{I}$ is an* optimal model *of $\mathcal{K}$ w.r.t. F if:*

- *$\mathcal{I} \models \mathcal{K}$,*
- *$v_F(\mathcal{I}) \neq \aleph_0$, and*
- *there exists no $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{K}$ and $v_F(\mathcal{J}) \leq v_F(\mathcal{I})$.*

If $\mathcal{I}$ is an optimal model of $\mathcal{K}$ w.r.t. F, we call $v_F(\mathcal{I})$ the optimal value *of $\mathcal{K}$ w.r.t. F.*

Example 5. *Going back to our KB $\mathcal{K}''$ from Example 4, we can now add the cost function $F = \sum rent[\top]$. The optimal models of $\mathcal{K}''$ are those that ensure that the total cost that our company has to pay for rent is as low as possible.*

We now define some natural reasoning tasks in optimal models of a KB $\mathcal{K}$ w.r.t a cost function F. One natural question we can ask for a given KB $\mathcal{K}$ and a cost function F for $\mathcal{K}$ is whether $\mathcal{K}$ has a optimal model w.r.t. F. Notice that $\mathcal{K}$ does not have an optimal model if the value of the models w.r.t F is finite but might grow unboundedly or if $\mathcal{K}$ only admits models whose value w.r.t. F is infinite.

OPTKBSAT	
Input:	A KB $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$ and a cost function F for $\mathcal{K}$.
Question:	Does $\mathcal{K}$ have a optimal model w.r.t. F?

Other standard DL reasoning tasks such as concept satisfiability and instance checking can be formulated for optimal models of a KB $\mathcal{K}$ w.r.t. a cost function F as follows.

OPTCONCEPTSAT
Input: A KB $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, a cost function F for $\mathcal{K}$, and a concept $B \in N_C(\mathcal{K})$.
Question: Is there an optimal model $\mathcal{I}$ of $\mathcal{K}$ w.r.t. F s.t. $B^{\mathcal{I}} \neq \emptyset$?

OPTINSTANCECHECK
Input: A KB $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, a cost function F for $\mathcal{K}$, and an assertion $B(a)$, where $B \in N_C(\mathcal{K})$, and $a \in N_I(\mathcal{K})$.
Question: Does $\mathcal{I} \models B(a)$ for every optimal model of $\mathcal{K}$ w.r.t. F?

The above reasoning tasks are straightforward adaptations of standard reasoning tasks to KBs equipped with cost functions. However, we can go beyond this and provide richer reasoning services, e.g. we can ask questions regarding the cost of some features in optimal models of $\mathcal{K}$. Recall the scenario of Example 2: we may want to know if it is *always* the case that in the models of our company's KB in which the cost of rent is minimal, the total capacity of the rented office space is greater than some number k. On the other hand, we could also consider a *brave* form of reasoning, and ask whether there is *at least one* model of our KB that minimizes the cost of rent and where the total capacity of office space is greater than some k.

We formalize such requests by means of *cost queries*. A cost query q is an expression of the form $F \circ w$, where F is a cost function defined as in (1), w is a non-negative rational, and $\circ \in \{<, \leq, =, \geq, >\}$. We say that a cost query q is true in an interpretation $\mathcal{I}$, in symbols $\mathcal{I} \models q$, if $v_F(\mathcal{I}) \circ w$ is true.

OPTCAUTIOUSCOSTQA
Input: A KB $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, a cost function F for $\mathcal{K}$, and a cost query q over the signature of $\mathcal{K}$.
Question: Does $\mathcal{I} \models q$ in all optimal models $\mathcal{I}$ of $\mathcal{K}$ w.r.t. F?

OPTBRAVECOSTQA
Input: A KB $\mathcal{K} = (\mathcal{T}, \Sigma, \alpha, \mathcal{A})$, a cost function F for $\mathcal{K}$, and a cost query q over the signature of $\mathcal{K}$.
Question: Is there an optimal model $\mathcal{I}$ of $\mathcal{K}$ w.r.t. F such that $\mathcal{I} \models q$?

4.1 Complexity of Optimal Model Reasoning

We now present tight complexity results for the reasoning problems introduced above. To this end, we observe that given a KB $\mathcal{K}$ and a cost function F for $\mathcal{K}$, we can define an exponentially sized enriched ILP whose optimal solutions correspond to optimal models of $\mathcal{K}$ w.r.t. F.

Proposition 5. *Given a KB $\mathcal{K}$ and a cost function F for $\mathcal{K}$, let $\mathcal{S}_\mathcal{K} = (V, \mathcal{E}, I)$ be the enriched system for $\mathcal{K}$ as described in Proposition 4. We can obtain from F and $\mathcal{K}$ (i) an EILP $\Pi_{\mathcal{K},F} = (V, \mathcal{E}, I, \min f)$ and (ii) an integer c that is at most exponential in the size of $\mathcal{K}$ such that the following hold:*

- *for every optimal solution S of $\Pi_{\mathcal{K},F}$, there is an optimal model $\mathcal{I}_S$ of $\mathcal{K}$ w.r.t. F such that for each tile τ the number of instances of τ in $\mathcal{I}$ is equal to $S_\mathcal{I}(\tau)$;*
- *for every optimal model $\mathcal{I}$ of $\mathcal{K}$ w.r.t. F there is an optimal solution of $\Pi_{\mathcal{K},F}$ such that for each tile τ the number of instances of τ in $\mathcal{I}$ is equal to $S_\mathcal{I}(\tau)$;*
- *the optimal value of $\Pi_{\mathcal{K},F}$ is equal to c times the optimal value of $\mathcal{K}$ w.r.t. F.*

The correctness of the proposition above mostly relies on the results from Proposition 4. We know that we can define an enriched system $\mathcal{S}_\mathcal{K}$ for $\mathcal{K}$ whose solutions correspond to models of $\mathcal{K}$ s.t. the variables of $\mathcal{S}_\mathcal{K}$ represent tiles for $\mathcal{K}$. As each tile carries the information about the feature values of their instances, we can easily reformulate the cost function F into an objective function $\min f$ for $\mathcal{S}_\mathcal{K}$. However, one issue that arises is that the weights of F and the feature values of tiles which are generally rational numbers now appear as coefficients in f. To ensure that we consider only integer coefficients, we multiply the coefficients by c which represents the least common multiple of their denominators.

We are now ready to address concept satisfiability in optimal models of KBs equipped with cost functions. We show that this problem can be solved in single exponential time, provided that access to an NP oracle is available.

Theorem 2. *The problem* OPTCONCEPTSAT *belongs to the class* $\text{EXPTIME}^{\text{NP}}$.

Proof. Take an instance of this problem consisting of a KB $\mathcal{K}$, a cost function F for $\mathcal{K}$, and a basic concept B. We first compute the optimal value of $\mathcal{K}$ w.r.t. F, if one exists. For this, it suffices to compute the optimal value of $\Pi_{\mathcal{K},F}$ and divide it by c, where $\Pi_{\mathcal{K},F} = (V, \mathcal{E}, I)$ is the EILP and c be the integer described in Proposition 5. From Proposition 2 we know that if there is a finite optimal value of $\Pi_{\mathcal{K},F}$, then there is one bounded by $m = \sum_{i=1}^{n} |c_i| \cdot |V|^2((|\mathcal{E}|+|I|)a)^{2(|\mathcal{E}|+|I|)+3}$, where a is the maximal coefficient in $\Pi_{\mathcal{K},F}$. It follows from the properties of the underlying enriched system $\mathcal{S}_\mathcal{K}$ (see Proposition 4) and the way that f was defined that m is at most double exponential in the size of $\mathcal{K}$. Hence, to compute the optimal value of $\mathcal{K}$ w.r.t. F we need to find in the interval $0 \dots m$ the integer j such that (a) $\mathcal{K}$ has a model $\mathcal{J}$ and $v_F(\mathcal{J}) = \frac{j}{c}$ and (b) $\mathcal{K}$ has no model $\mathcal{J}$ and $v_F(\mathcal{J}) < \frac{j}{c}$. For this, we can perform a binary search in $0 \dots m$ using an NP oracle. After finding j, we make an NP oracle call to check if $\mathcal{K}$ has a model $\mathcal{J}$ with $v_F(\mathcal{J}) = \frac{j}{c}$ and $B^{\mathcal{J}} \neq \emptyset$. This is possible by checking whether there is a solution to the enriched system for $\mathcal{K}$ with two additional inequalities encoding that (i) the value of solutions w.r.t. F is j and (ii) the satisfaction of B. Finding the correct j takes at most $\lceil log(m) \rceil$ oracle calls, which results in a procedure that runs in single exponential time, assuming an access to an NP oracle. In each call, we supply the oracle with an (exponentially large) integer inequality system to decide the existence of a model $\mathcal{J}$ of $\mathcal{K}$ with $v_F(\mathcal{J}) \leq \frac{j}{c}$, where j is an integer (a possibly exponentially long string in binary) that corresponds to the active middle element in the binary search. □

We can show that the above complexity bound is tight.

Proposition 6. OPTCONCEPTSAT *is* EXPTIME$^{\mathsf{NP}}$*-hard.*

Proof. We consider non-deterministic exponential time *metric Turing machines*. Each such a machine M takes as input a word $x \in \{0,1\}^*$ and at the end of its computation outputs a word $M(x) \in \{0,1\}^*$, which is seen as an integer encoded in binary. The machine M runs in exponential time, i.e., computing $M(x)$ takes at most $2^{p(|x|)}$ steps, where p is a fixed polynomial. We let $min^M(x)$ denote the smallest value of $M(x)$ over all runs of M on x. This model is an adaptation of the one in (Krentel 1988), where the author introduced non-deterministic *polynomial* time metric Turing machines. Next, consider the following problem:

NEXPMINEVEN	
Input:	A description of a metric TM M whose run time is bounded by an exponential, and a word $x \in \{0,1\}^*$.
Question:	Is $min^M(x)$ an even number?

By adapting the proof of Theorem 3.1 in (Krentel 1988), it can be seen that NEXPMINEVEN is EXPTIME$^{\mathsf{NP}}$-hard. Once this result is established (see Appendix), the lower bound for OPTCONCEPTSAT can be seen via a polynomial time reduction from NEXPMINEVEN. We only provide a high level description this reduction, because it uses standard ingredients. Asssume an instance (M, x) of NEXPMINEVEN. We know that M terminates on x within some $n = 2^{p(|x|)}$ steps, where p is a polynomial. Moreover, in every run on x, the machine M produces an integer value of with no more than n bits. Here are the main components of a KB $\mathcal{K}$ that captures the possible computation of M on x:

- We have inclusions that enforce a $2^n \times 2^n$ grid. The inclusions to create such a grid are standard (Tobies 2000). Furthermore, using special concepts, we can ensure that each position (x, y) in the grid is encoded at a unique domain element (in binary). We use roles s and t to navigate this grid, where s stands for *space* and t stands for *time*.
- The inclusion to simulate the run of a non-deterministic Turing machine are also standard, e.g. (Krötzsch, Rudolph, and Hitzler 2013). Every model of the KB will capture one non-determinstic computation of M on x, which will be represented in the above mentioned grid.
- We can make sure that the output number that M produces on x is captured at the grid's border that corresponds to the instantaneous description of M at time 2^n. In particular, in a model $\mathcal{I}$ this border contains domain objects $(o_1, o_2) \in s^{\mathcal{I}}, (o_2, o_3) \in s^{\mathcal{I}}, \ldots, (o_{2^n-1}, o_{2^n}) \in s^{\mathcal{I}}$ related by the role s. The output number is coded in binary using concept names Zero and One. We can assume w.l.o.g. that o_1 and o_{2^n} store the least significant and the most significant bits of the output number, respectively.
- We need a way to "convert" the number written on the grid's border into a representation that can be used for minimization. For this, we generate at each o_i, satisfying the concept One, a tree with exactly 2^{i-1} leaves, this is also standard in our logic (Baader et al. 2017). Each leaf can be labeled with a special concept Leaf. We can now count the leaves of each tree by introducing a feature $count$ and defining an annotation function $\alpha(\mathsf{Leaf}, count) = \{1\}$. We can then consider the cost function $F := \sum count[\mathsf{Leaf}]$. Our goal is to select models $\mathcal{I}$ where the value $v_F(\mathcal{I})$ is minimal.
- The final step to devise the goal concept Goal, which should be non-empty in a model $\mathcal{I}$ iff $o_1 \in \mathsf{Zero}^{\mathcal{I}}$. This is again easy to do in our logic, as we can 'detect' o_1 by using the concepts corresponding to the binary encoding of the coordinates of o_1 and check if Zero is satisfied. □

Based on this result, we can now state tight complexity bounds for the rest of our reasoning tasks

Theorem 3. *The following complexity results are correct:*

- OPTKBSAT *is* NEXPTIME*-complete;*
- OPTINSTANCECHECK, OPTCAUTIOUSCOSTQA *and* OPTBRAVECOSTQA *are* EXPTIME$^{\mathsf{NP}}$*-complete.*

Proof. Given a KB $\mathcal{K}$ and a cost function F, to check whether $\mathcal{K}$ has an optimal model w.r.t to F, it suffices to check whether there exists at least one model $\mathcal{I}$ of $\mathcal{K}$ s.t. $v_F(\mathcal{I})$ is finite. This can be done by checking whether the enriched system that encodes the satisfiability for $\mathcal{K}$ has a solution in which all variables that correspond to the tiles that contribute to $v_F(\mathcal{I})$ take on finite values. The last check is known to reduce to regular feasibility problem of ILP and is possible in NP. Due to Proposition 4, the NExpTime upper bound follows. The matching lower bound comes from the NEXPTIME-hardness of KB satisfiability in $\mathcal{ALCHOIQ}$.

The upper bound for OPTINSTANCECHECK follows from the reduction to OPTCONCEPTSAT, while we can show the same upper bound for OPTCAUTIOUSCQA, OPTBRAVECQA by straightforward modifications of the algorithm in Theorem 2, where the last NP oracle call checks for the (un)satisfaction of the given cost query instead of the concept. The matching lower bounds are obtained by reduction from OPTCONCEPTSAT. □

Optimal models and maximization. So far, we considered as optimal models of a KB $\mathcal{K}$ w.r.t. a given cost function F those models whose value w.r.t. F is minimal. We can analogously define optimal models as those that *maximize* the value of F. The above EXPTIME$^{\mathsf{NP}}$-completeness results carry over to the maximization setting. However, this does not apply to OPTKBSAT, where the previous NEXPTIME-completeness was based on finding a model of a KB with some arbitrary finite cost (which in turn was sufficient to guarantee the existence of a model with a minimal cost). In the case of maximization, solving OPTKBSAT involves two tests: (a) checking that the input KB is consistent, and (b) checking that there is a rational number b such that, in all models of the KB, the cost function yields a value that is bounded by b. The first task is trivially in NEXPTIME, while the second task is in co-NEXPTIME. The latter follows from Propositions 5 and 2, since deciding (b) amounts to checking whether an exponentially sized system has a finite optimal value. Checking (b) is also co-NEXPTIME-hard, which can be shown by a reduction from the predicate boundedness problem in $\mathcal{ALCHOIQ}$ with closed predicates (Lukumbuzya and Šimkus 2021).

5 Related Work and Discussion

Concrete domains. Our way of adding numeric features to DLs resembles *concrete domains*, which have been extensively studied since the early days of DLs, e.g., (Baader and Hanschke 1991; Lutz and Milicic 2007; Borgwardt, De Bortoli, and Koopmann 2024). Reasoning with concrete domains quickly becomes undecidable, especially in the presence of general TBoxes (Lutz 2002), unless the concrete domain is severely restricted, for example by limiting the arity of the predicates and imposing some form of compactness, as in ω-admissibility (Baader and De Bortoli 2024). Integer numbers and addition over them—the basics of numeric reasoning—are no-gos in the concrete domain literature; addition almost always causes undecidability (Lutz 2002), while admissibility conditions typically fail for the integers (Baader and Rydval 2022). We can support both because other restrictions are imposed. Our feature annotations are discrete, finite and positive, and the induced numerical equations are finite and can be effectively solved. Moreover, our numeric restrictions are on direct neighbors only, similar to a few older works on *path-free* concrete domains (Pan and Horrocks 2002; Haarslev, Möller, and Wessel 2001). This is fundamental to our upper bounds, allowing us to search only for models where there are only exponentially many objects that differ in terms of both their relational and numeric types.

DLs with cardinality restrictions. We have shown that our numerical features can simulate the traditional *local* number restrictions ($\mathcal{Q}$), which count the neighbors of a node. The related *cardinality restrictions*, which globally count the objects participating in concepts, have also been around for at least two decades (Tobies 2000). $\mathcal{ALCHOIQ}$ with closed predicates is already expressive enough to simulate global cardinality restrictions and even some properties that are not first-order definable, like testing if the cardinality of a concept is even (Lukumbuzya, Ortiz, and Šimkus 2024; Lukumbuzya 2024). There has been significant interest in the last few years in extending both types of cardinality restrictions. We now have powerful extensions of expressive description logics with *extended cardinality constraints* that may even use the quantifier-free fragment of Boolean Algebra with Presburger Arithmetic (QFBAPA), and a relatively complete understanding of their complexity and expressive power (Baader, Bednarczyk, and Rudolph 2020; Baader and De Bortoli 2020; Bednarczyk and Fiuk 2022). These formalisms are orthogonal to our extension; reconciling the differences seems challenging, but worth pursuing. Additionally to the rich cardinality constraints, our logic has a form of concrete domains. The combination of these two types of numeric reasoning, despite being very natural, has not received much attention. We are aware of only one very recent exception (Baader et al. 2025).

Reasoning about preferred models. To our knowledge, the reasoning tasks over optimal models introduced in Section 4 had not been studied in the context of DLs. Some of the underlying intuitions are reminiscent of *circumscription* (Bonatti, Lutz, and Wolter 2009; Di Stefano, Ortiz, and Šimkus 2023), which is also a form of *preferred models semantics*. In our approach, models are discarded in favor of other models whose cost value is lower, while in circumscription, one discards models where the extension of some predicates is not minimal. Our formalism allows for a form of predicate minimization: we can introduce a feature f_A for each concept name A that is meant to be minimized, set $\alpha(A, f_A) = \{1\}$ and choose models that minimize the cost of f_A. We cannot, however, simulate complex circumscription patterns and enforce global minimization over a fixed domain, and our formalism is, in general, computationally simpler than circumscription.

6 Conclusion and Future Work

In this paper, we proposed an extension of classical DLs with numeric features and with constraints on the aggregated feature values in an object's neighborhood, expressed via a new concept constructor, called *neighborhood restrictions*. We illustrated how our formalism provides a natural and easy-to-use tool for modeling and reasoning in complex domains. Imposing natural restrictions on feature values and weights, standard reasoning tasks in $\mathcal{ALCHIOQ}$ with closed predicates and neighborhood restrictions are not computationally harder than in classical $\mathcal{ALCHIOQ}$. We also studied non-standard reasoning problems inspired by Constraint Programming (CP). By means of *cost-functions*, we defined the notion of *optimal models* for a KB. We showed that checking the existence of optimal models does not increase in complexity compared to standard reasoning tasks. Furthermore, we introduced *cost-queries*, which retrieve feature values and compare them to a given weight. We characterized the complexity of brave and cautious entailment of cost-queries and standard reasoning tasks in optimal models.

The integration of CP techniques in KR formalisms received remarkable attention in recent years, e.g. in Answer Set Programming (ASP) with its powerful solver *clingcon* (Gebser, Ostrowski, and Schaub 2009; Ostrowski and Schaub 2012), where support for CP constructs was introduced. A crucial difficulty in combining CP techniques in DLs is given by the absence of restrictions on the domain sizes, which can possibly be infinite. Intuitively, the numeric constraints we introduced can be seen as a succinct specification of an infinite collection of instances of integer linear programming (ILP) problems. An important contribution of this work is that we can effectively reason over these succinct systems of constraints by resorting to ILP techniques.

An implementation of our reasoning services is left for future work. Another non-trivial direction is to relax the restrictions on features and numeric constraints, e.g., by allowing negative values for features and weights.

We also see our approach as a promising hosting formalism for inconsistency-tolerant reasoning, using features to replace inconsistencies with an 'error' cost and selecting models where the error costs are minimal. We aim to explore whether this yields a meaningful semantics, and how it compares with recent cost-based semantics for inconsistent KBs (Bienvenu, Bourgaux, and Jean 2024). The latter application may require extending our formalism with multi-objective functions, expressing priorities and relaxing the notion of optimal models.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) projects PIN8884924, P30873 and 10.55776/COE12.

References

Baader, F., and De Bortoli, F. 2020. Description Logics That Count, and What They Can and Cannot Count. In Kovacs, L.; Korovin, K.; and Reger, G., eds., *ANDREI-60. Automated New-era Deductive Reasoning Event in Iberia*, volume 68 of *EPiC Series in Computing*, 1–25. EasyChair.

Baader, F., and De Bortoli, F. 2024. The Abstract Expressive Power of First-Order and Description Logics with Concrete Domains. In Hong, J., and Park, J. W., eds., *Proc. of the 39th ACM/SIGAPP Symposium on Applied Computing, (SAC)*, 754–761. ACM.

Baader, F., and Hanschke, P. 1991. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proc. of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, 452–457. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Baader, F., and Rydval, J. 2022. Using Model Theory to Find Decidable and Tractable Description Logics with Concrete Domains. *J. Autom. Reason.* 66(3):357–407.

Baader, F.; Bednarczyk, B.; and Rudolph, S. 2020. Satisfiability and query answering in description logics with global and local cardinality constraints. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and Lang, J., eds., *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 616–623. IOS Press.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Baader, F.; Borgwardt, S.; De Bortoli, F.; and Koopmann, P. 2025. Concrete domains meet expressive cardinality restrictions in description logics (extended version). *CoRR* abs/2505.21103.

Bednarczyk, B., and Fiuk, O. 2022. Presburger Büchi Tree Automata with Applications to Logics with Expressive Counting. In Ciabattoni, A.; Pimentel, E.; and de Queiroz, R. J. G. B., eds., *Proc. of the 28th International Workshop on Logic, Language, Information, and Computation (WoLLIC)*, volume 13468 of *Lecture Notes in Computer Science*, 295–308. Springer.

Bienvenu, M.; Bourgaux, C.; and Jean, R. 2024. Cost-Based Semantics for Querying Inconsistent Weighted Knowledge Bases. In Marquis, P.; Ortiz, M.; and Pagnucco, M., eds., *Proc. of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Bonatti, P. A.; Lutz, C.; and Wolter, F. 2009. The Complexity of Circumscription in DLs. *J. Artif. Intell. Res.* 35:717–773.

Borgwardt, S.; De Bortoli, F.; and Koopmann, P. 2024. The Precise Complexity of Reasoning in $\mathcal{ALC}$ with ω-Admissible Concrete Domains. In Giordano, L.; Jung, J. C.; and Ozaki, A., eds., *Proc. of the 37th International Workshop on Description Logics (DL)*, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Demri, S., and Quaas, K. 2023. First Steps Towards Taming Description Logics with Strings. In Gaggl, S. A.; Martinez, M. V.; and Ortiz, M., eds., *Proc. of the 18th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 14281 of *Lecture Notes in Computer Science*, 322–337. Springer.

Di Stefano, F.; Ortiz, M.; and Šimkus, M. 2023. Description Logics with Pointwise Circumscription. In *Proc. of the 32th International Joint Conference on Artificial Intelligence, (IJCAI)*, 3167–3175.

Gebser, M.; Ostrowski, M.; and Schaub, T. 2009. Constraint answer set solving. In Hill, P. M., and Warren, D. S., eds., *Logic Programming*, 235–249. Berlin, Heidelberg: Springer Berlin Heidelberg.

Gogacz, T.; Gutiérrez-Basulto, V.; Ibáñez-García, Y. A.; Murlak, F.; Ortiz, M.; and Šimkus, M. 2019. Ontology Focusing: Knowledge-Enriched Databases on Demand. *arXiv preprint arXiv:1904.00195*.

Haarslev, V.; Möller, R.; and Wessel, M. 2001. The Description Logic $\mathcal{ALCNH}_{\mathrm{R+}}$ Extended with Concrete Domains: A Practically Motivated Approach. In Goré, R.; Leitsch, A.; and Nipkow, T., eds., *Proc of the 1st International Joint Conference on Automated Reasoning (IJCAR)*, volume 2083 of *Lecture Notes in Computer Science*, 29–44. Springer.

Krentel, M. W. 1988. The Complexity of Optimization Problems. *Journal of Computer and System Sciences* 36(3):490–509.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2013. Complexities of Horn Description Logics. *ACM Trans. Comput. Logic* 14(1).

Labai, N.; Ortiz, M.; and Šimkus, M. 2020. An ExpTime Upper Bound for $\mathcal{ALC}$ with Integers. In Calvanese, D.; Erdem, E.; and Thielscher, M., eds., *Proc. of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 614–623.

Lukumbuzya, S., and Šimkus, M. 2021. Bounded Predicates in Description Logics with Counting. In Zhou, Z., ed., *Proc. of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 1966–1972. ijcai.org.

Lukumbuzya, S.; Ortiz, M.; and Šimkus, M. 2024. Datalog Rewritability and Data Complexity of $\mathcal{ALCHOIQ}$ with Closed Predicates. *Artif. Intell.* 330:104099.

Lukumbuzya, S. 2024. *Reasoning in Very Expressive Description Logics with Varying Information Completeness*. Ph.D. Dissertation, TU Wien. https://doi.org/10.34726/hss.2025.123260.

Lutz, C., and Milicic, M. 2007. A Tableau Algorithm for Description Logics with Concrete Domains and General TBoxes. *J. Autom. Reason.* 38(1-3):227–259.

Lutz, C. 2002. *The Complexity of Description Logics with Concrete Domains*. Ph.D. Dissertation, RWTH Aachen University, Germany.

Ostrowski, M., and Schaub, T. 2012. ASP modulo CSP: The clingcon System. *Theory and Practice of Logic Programming* 12(4–5):485–503.

Pan, J. Z., and Horrocks, I. 2002. Reasoning in the $\mathcal{SHOQ}(\mathrm{D_n})$ Description Logic. In Horrocks, I., and Tessaris, S., eds., *Proc. of the 2002 International Workshop on Description Logics (DL)*, volume 53 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Papadimitriou, C. H. 1981. On the Complexity of Integer Programming. *J. ACM* 28(4):765–768.

Tobies, S. 2000. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *J. Artif. Intell. Res.* 12:199–217.