Description Logic with part restrictions: PSPACE-complete expressiveness

Mitko Yanchev

Faculty of Mathematics and Informatics, Sofia University, Bulgaria
yanchev@fmi.uni-sofia.bg

Abstract. In this paper syntactic objects—concept constructors, called *part restrictions*—are considered in Description Logics (DLs). Being able to convey statements about a part of a set of successors, part restrictions essentially enrich the expressive capabilities of DLs. An extension of DL $\mathcal{ALCQR}$ with part restrictions is examined, and its PSPACE completeness is proven, what shows that the new expressiveness brings no extra cost to the complexity of reasoning in $\mathcal{ALCQR}$. The proof uses completion calculus based on tableaux technique.

Keywords: Description Logics, part restrictions, complexity, PSPACE

1 Introduction

Description Logics are logical formalism widely used in knowledge-based systems. They both explicitly represent knowledge in form of taxonomy, and infer new knowledge out of the presented structure by means of a specialized inference engine. The representation language, called *concept language*, comprises expressions with only unary and binary predicates, called *concepts* and *roles*, respectively. Concepts describe sets of objects in the domain, while roles represent links between objects. Examples of these are PERSON, MALE, HAS_CHILD.

Concept languages differ mainly in the constructors adopted for building complex concepts and roles, and they are compared with respect to their expressiveness, as well as with respect to the complexity of reasoning in them. DL $\mathcal{ALCQR}$ adopts the most natural for an expressive DL concept constructors: negation, intersection, union, universal and existential role quantifications, and *qualifying number restrictions*, together with the constructor for *intersection of roles*. Qualifying number restrictions are 'counting' constructors analogous to *grading* operators in modal logics. Tobies proves [1] that the satisfiability problem for the formulae in the language of Graded Modal Logic, a syntactic analogue of $\mathcal{ALCQ}$, in which only independent roles are adopted, is PSPACE-complete.

We consider concept constructors, which we call *part restrictions*, capable to distinguish a part of a set of successors. These constructors are analogues of the modal operators for *rational grading* [3] which generalize the majority operators [2]. They are $MrR.C$ and (the dual) $WrR.C$, where r is a rational number in $(0, 1)$, R is a role, and C is a concept. The intended meaning of $MrR.C$ is 'More

than r-part of all R-successors of the current object belongs to (the interpretation of) C'. Part restrictions essentially enrich the expressive capabilities of DLs. They can express, for example, notions like qualified majority:

$$M_{2/3}\, voted.Yes$$

Presburger constraints in the language of Presburger Modal Logic (PML) capture both integer and rational grading, and have rich expressiveness. Demri and Lugiez prove [4] that the satisfiability problem for the language of PML, a many-relational modal language with independent relations, is PSPACE-complete, thus strengthening the main result in [1], and answering the same question about the language with majority operators.

The rational grading operators are expressible by the presburger constraints. Thus, the PSPACE completeness of satisfiability for the corresponding modal language with rational grading is a consequence of that for the PML language. On the other hand, the presence of separate integer and rational grading constructors also proves beneficial. Together with the use of specific technique for exploring the complexity of rational grading it allows following a common way for obtaining complexity results as in less, so in more expressive languages with rational grading. In particular, complexity results for the satisfiability—polynomial, NP, and co-NP—concerning a range of description logics from the $\mathcal{ALC}$-family with part restrictions added, are obtained (Yanchev, 2012, 2013).

Now we consider the DL $\mathcal{ALCQPR}$, the extension of $\mathcal{ALCQR}$ with part restrictions. We use completion calculus based on tableau technique to prove that the reasoning complexity in $\mathcal{ALCQPR}$ is still in PSPACE.

2 Syntax and semantics of $\mathcal{ALCQPR}$. Inference problems

Let $\mathbf{A}$ be a set of *atomic concepts*, denoted by A, $\mathbf{R}$ be a set of *atomic roles*, denoted by P, and $\mathbf{Q}$ be a set of rational numbers in $(0, 1)$. $\mathbf{R}^{\sqcap}$ denotes the set of all finite intersections of atomic roles, called *roles*. The *Description Logic* $\mathcal{ALCQPR}$ is the smallest set of *concepts*, obtained inductively using the following concept constructors, and according to the following syntactic rules: it contains every atomic concept, the *universal concept* $\top$ and the *empty concept* $\bot$, and, if it contains the concepts C and D, R is a role, i.e. $R = P_1 \sqcap ... \sqcap P_k$, for some $k \geq 1$, $n \geq 0$, and $r \in \mathbf{Q}$, then it contains also:

$\neg C \mid$	(negation)
$C \sqcap D \mid$	(intersection)
$C \sqcup D \mid$	(union)
$\forall R.C \mid$	(universal role quantification)
$\exists R.C \mid$	(existential role quantification)
$\geqslant nR.C$ and $\leqslant nR.C \mid$	(qualifying number restrictions)
$MrR.C$ and $WrR.C$	(part restrictions)

An *interpretation* $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty set $\triangle^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$, which maps every concept to a subset of $\triangle^{\mathcal{I}}$

and every role to a subset of $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$. For the 'old' constructors the mapping is the usual (see, e.g. in [6]). For part restrictions, for any concept C, role R, and rational number $r \in \mathbf{Q}$ the properties that follow hold. $R^{\mathcal{I}}(a)$ denotes the set of $R^{\mathcal{I}}$-successors of a, and $R^{\mathcal{I}}(a, C)$ denotes the set of $R^{\mathcal{I}}$-successors of a which are in $C^{\mathcal{I}}$, i.e. the set $\{b \mid (a, b) \in R^{\mathcal{I}} \ \& \ b \in C^{\mathcal{I}}\}$.

$$(MrR.C)^{\mathcal{I}} = \{a \in \triangle^{\mathcal{I}} : |R^{\mathcal{I}}(a, C^{\mathcal{I}})| > r.|R^{\mathcal{I}}(a)|\}$$
$$(WrR.C)^{\mathcal{I}} = \{a \in \triangle^{\mathcal{I}} : |R^{\mathcal{I}}(a, (\neg C)^{\mathcal{I}})| \leq r.|R^{\mathcal{I}}(a)|\}$$

The interpretation function is fully determined by the way it interprets the atomic concepts and atomic roles.

A concept C is *satisfiable* if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \varnothing$. Such an interpretation is called a *model* for that concept. For $a \in C^{\mathcal{I}}$ we say that a *satisfies* C, while $a \in \triangle^{\mathcal{I}} \backslash C^{\mathcal{I}}$ *refuses* C.

Thus, an object $a \in \triangle^{\mathcal{I}}$ is in $(MrR.C)^{\mathcal{I}}$ if (strictly) greater than r part of $R^{\mathcal{I}}$-successors of a satisfies C, and a is in $(WrR.C)^{\mathcal{I}}$ if no greater than r part of $R^{\mathcal{I}}$-successors of a refuses C.

A concept C is *subsumed* by a concept D, denoted $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for any interpretation $\mathcal{I}$.

Checking if a concept is subsumed by another concept is the most general reasoning task in DLs, as all other inference problems in DLs can be reduced to it. In particular, the satisfiability can be reduced to subsumption, as a concept C is unsatisfiable iff $C \sqsubseteq \bot$. So, the reasoning complexity of a DL is the complexity of checking the subsumption between its concepts.

From the other side, $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable. Thus, in the presence of negation of an arbitrary concept in a DL, what is the case with $\mathcal{ALCQPR}$, checking the (un)satisfiability becomes as complex as checking the subsumption.

In the following we will only consider concepts in the *negation normal form* (NNF) in which negations have been moved inwards, and can appear only in front of the atomic concepts. The NNF of $\neg C$ we denote by $\sim C$. Note that the NNF of a concept can always be obtained in linear time and space.

3 A calculus for checking the satisfiability

$\mathcal{ALCQPR}$ is PSPACE-hard, as it is an extension of $\mathcal{ALC}$, which is PSPACE-complete [5]. We prove the PSPACE completeness of $\mathcal{ALCQPR}$ by presenting a PSPACE algorithm that decides the satisfiability of its concepts. The algorithm tries to build a model for a concept by manipulating *sets of constraints* induced by that concept, with the help of so-called *completion rules*. This is the well-known tableau technique for checking the satisfiability, already used to prove decidability and complexity results for a range of DLs ([5], [6]) and modal logics ([1]). We base our algorithm on the one presented in [1] for a many-relational graded modal language adopting intersection of relations. We make use of a specific technique, reflecting the presence of part restrictions in the language, to design the extended set of completion rules.

Definition 1. Let $\mathcal{V}$ be a set of syntactic variables. A *constraint* is a syntactic object, having one of the forms

$$x : C,\ \ Pxy,$$

where $x, y \in \mathcal{V}$, C is a concept, and P is an atomic role.
A *constraint system* (c.s.) S is a finite set of constraints. For $x : C \in S$ we say that C is *assigned* to x in S, and for $Pxy \in S$—that y is a P*-successor* of x in S.

Let $R = P_1 \sqcap ... \sqcap P_k$. We use the notations $P \in_n R$ (n is for 'as a $n$ame'), and $Q \subseteq_n R$, if P is P_i for some i, $1 \leq i \leq k$, and Q is an intersection of atomic roles amongst $P_1, ..., P_k$. With the same meaning we use the notations $P \in_n \{P_1, ..., P_k\}$ and $Q \subseteq_n \{P_1, ..., P_k\}$. We also use as an abbreviation the notation $Rxy \in S$, if $\{P_1xy, ..., P_kxy\} \subseteq S$, and we say that y is an R-successor of x in S, without having defined explicitly constraints with arbitrary roles.

We denote with $R^S(x)$ the set of R-successors of x in S, and with $R^S(x, C)$—the set of R-successors of x in S with C assigned in S, i.e. $\{y | \{Rxy, y : C\} \subseteq S\}$.

A c.s. S is said to contain a *clash* (an obvious contradiction) if for some atomic concept A, a concept C, $n \geq 0$, and $r \in \mathbf{Q}$ any of the following holds:

$CL0$. $\{x : \bot\} \subseteq S$
$CL1$. $\{x : A,\ x : \neg A\} \subseteq S$
$CL2$. $\{x : \leqslant nR.C\} \subseteq S$ *and* $|R^S(x, C)| > n$
$CL3$. $\{x : MrR.C\} \subseteq S$ *and* $|R^S(x, C)| \leq r.|R^S(x)|$
$CL4$. $\{x : WrR.C\} \subseteq S$ *and* $|R^S(x, \sim C)| > r.|R^S(x)|$

Otherwise it is called *clash-free*.

A c.s. S is called *complete* if none of the *completion rules* can be applied to it. In Figure 1 are presented only the generating ones from the set of completion rules. The non-generating ones are $\rightarrow_\sqcap$-, $\rightarrow_\sqcup$, and $\rightarrow_\forall$-rule, and have the usual form, see e.g. [6]. Note that we do not need, as in [1], the $\rightarrow_\leq$-rule.

$\rightarrow_\exists$-rule: If 1. $x : \exists R.C \in S$, and
2. there is no z such that $\{Rxz, z : C\} \subseteq S$,
Then $S \rightarrow_\exists S \cup S_{inc}$

$\rightarrow_\geq$-rule: If 1. $x : \geq nR.C \in S$, and
2. $|R^S(x, C)| < n$,
Then $S \rightarrow_\geq S \cup S_{inc}$

$\rightarrow_M$-rule: If 1. $x : MrR.C \in S$,
2. $|R^S(x, C)| \leq r.|R^S(x)|$, and
3. $|R^S(x)| < BAS_x$
Then $S \rightarrow_M S \cup S_{inc}$

$\rightarrow_W$-rule: If 1. $x : WrR.C \in S$,
2. $|R^S(x, \sim C)| > r.|R^S(x)|$, and
3. $|R^S(x)| < BAS_x$
Then $S \rightarrow_W S \cup S_{inc}$

Fig. 1. The generating completion rules

To test the satisfiability of a concept C_0, the algorithm starts with the c.s. $\{x_0 : C_0\}$, and successively applies the completion rules, generating new c.s. until a complete c.s. is obtained, or an instance of $CL0$–$CL2$ occurs in a c.s. The algorithm answers 'C_0 *is satisfiable*' if the rules can be applied in a way that yields a complete and clash-free c.s.

The algorithm makes non-deterministic choice in both which rule to be applied, and which concept to be assigned (as in the $\rightarrow_{\sqcup}$-rule, so in all generating rules). However, as we aim a PSPACE algorithm, the non-determinism makes no problem due to Savitch's theorem which states that deterministic and non-deterministic polynomial space coincide.

3.1 Completion rules. Rules application strategy

The interaction between the constructors is such that we can keep the rules from the 'integer' case in principle with no changes. These are $\rightarrow_{\sqcap}$-, $\rightarrow_{\sqcup}$-, $\rightarrow_{\forall}$-, $\rightarrow_{\exists}$-, and $\rightarrow_{\geq}$-rule. $\rightarrow_M$- and $\rightarrow_W$-rule deal with part restrictions.

We refer to the rules which add constraints with new variables to the c.s. as *generating rules*. These are $\rightarrow_{\exists}$-, $\rightarrow_{\geq}$-, $\rightarrow_M$-, and $\rightarrow_W$-rule.

The set of constraints added by any of the generating rules is formed in one and same way. Let $\boxdot = \exists \,|\, \forall \,|\, \geq m \,|\, \leq m \,|M_q|W_q$, and $\boxplus = \exists \,|\, \geq m \,|M_q|W_q$, for arbitrary $m \geq 0$, and $q \in \mathbf{Q}$, i.e. $\boxdot$ is used to refer to the constructors, talking about the successors (we will call them *successor constructors*), while $\boxplus$ is used to refer to those of them which trigger the generating rules (so, we will call them *generating constructors*). Using this notation, let the triggering constraint be $x : \boxplus R.C$. Then the set of constraints to be added by the rule to the c.s. S is:

$$S_{inc} = \{y : C\} \cup S' \cup S'',$$

where y is a new variable (what makes the rule a generating one).

$$S' = \{y : E_1, ..., y : E_l\},$$

where $E_1, ..., E_l$ are obtained as follows. Let $\{D_1, ..., D_l\} = \{D \mid x : \boxdot Q.D \in S\}$, i.e. $D_1, ..., D_l$ are all concepts, preceded by a 'successor' constructor, occurring in constraints at x in S. Then each E_i is chosen non-deterministically to be either D_i, or $\sim D_i$, i.e. $E_i \in \{D_i, \sim D_i\},\ i = 1, ..., l$.

$$S'' = \{P_1 xy, ..., P_k xy\},$$

where $P_1, ..., P_k$ are chosen non-deterministically such that $R \subseteq_n \{P_1, ..., P_k\} \subseteq \mathbf{R}_{C_0}$. With $\mathbf{R}_C$ we denote the set of atomic roles occurring in the concept C; also, $\mathbf{R}_C^{\sqcap}$ will denote the set of roles in C.

Border amount of successors for the current variable x (BAS_x) is used to ensure inapplicability of 'part' rules after a given moment. It is a key notion for the generation process termination, and will be discussed in details later.

We keep as *rules application strategy* the requirement from [1] all possible applications of non-generating rules (i.e. $\rightarrow_{\sqcap}$-, $\rightarrow_{\sqcup}$-, and $\rightarrow_{\forall}$-rule) to be done first, thus ensuring all possible constraints of the form $x : \boxdot R.C$ to be added in the c.s. before a generating rule is applied. Also, instead of generating all successors of a variable at the same time, rules generate them one-by-one, and non-deterministically assign formulae to a successor at its generation.

3.2 The clashes with part restrictions

Part restrictions talk about no exact quantities, but ratios, what makes $CL3$ and $CL4$, which are also conditions for applicability of part rules, dynamic. Applied consecutively, part rules can 'repair' one clash, and, at the same time, provoke another. Thus, instances of $CL3$ and $CL4$ can appear and disappear infinitely during c.s. generation, even if the initial concept is satisfiable. So we have to take special care to ensure the termination of part rules application, without leaving avoidable clashes in the c.s. That turns out to be the main difficulty in designing the algorithm. We overcome it by proving that if it is possible to unfold part restrictions at a given variable avoiding *simultaneously* both kinds of clashes, it can be done within some number of successors. As clashes are always connected with a single variable, that is enough to guarantee the termination. The following subsection presents the technique in details.

3.3 Counteracting constraints. Clusters

In a c.s. generated from an initial c.s. having the form $\{x : C\}$ an instances of $CL3$ and $CL4$ can appear only if two contradicting concepts have been assigned to one and the same variable. We start our analysis with the simplest case when at a variable x we have only constraints with part restrictions, and all they are with one and the same role R, and with subconcepts which are either a fixed concept C, or its negation $\sim C$. All such constraints form the set:

$$\{x : Mr_1R.C,\ x : Mr_2R.\sim C,\ x : Wr_3R.C,\ x : Wr_4R.\sim C\} \quad (1)$$

We call the subset of (1) which is in a c.s. S a *cluster of R and C at x in S*. We denote it $Cl_x^S(R, C)$.

Definition 2. A *constraint with part restriction is S-satisfied*, where S is a c.s., if the inequality 2. from the corresponding completion rule fails in S.

A *cluster is S-satisfied* if all constraints in it are S-satisfied.

A *cluster is c.s.-satisfiable* if it can be S-satisfied for some c.s. S.

In fact, we can have in (1) more than one constraint of any of the four kinds. But note that, if $x : MrR.C$ is S-satisfied, then that is the case with $x : Mr'R.C$, for any $r' \leq r$. So, we can take r_1 and r_2 to be the maximums, and, by analogues reasons, r_3 and r_4 to be the minimums of the rationals in constraints of the corresponding kinds. Thus we obtain the upper, representative for the c.s.-satisfiability of all part constraints, set with only four ones.

The idea behind c.s.-satisfiability is that if a cluster, and, more general, the set of all part constraints at a given variable is c.s.-satisfiable, then a c.s. without clashes with these constraints can be non-deterministically generated, while the rules become inapplicable to these constraints. So, concerning part constraints c.s.-satisfiability is sufficient condition for obtaining a clash-free complete c.s.

Our next observation is that both $(x:)Mr_1R.C$ and $(x:)Wr_3R.C$ act in one and the same direction concerning c.s. generation, as the former forces the addition of enough R-successors of x with C assigned, and the latter limits the

number of R-successors of x with $\sim C$ assigned. The situation is analogues with $Mr_2R.\sim C$ and $Wr_4R.\sim C$ with respect to $\sim C$. At that time as $Mr_1R.C$, so $Wr_3R.C$ counteract with any of $Mr_2R.\sim C$ and $Wr_4R.\sim C$. This leads to two main possibilities for the cluster $Cl_x^S(R,C)$:

A. The cluster contains only constraints, acting in one and the same direction (or just a single constraint)—we call it *cluster of type* A, or A-cluster. In the absence of counteracting constraints these clusters are always c.s.-satisfiable.

B. The cluster contains at least two counteracting constraints—we call it *cluster of type* B, or B-cluster.

In order to c.s. generation process to be able to c.s.-satisfy a B-cluster, and to avoid clashes, the next inequalities between the rationals in the cluster must be fulfilled—follows directly from the definition of c.s.-satisfiability:

1° $r_1 + r_2 < 1$,
2° $r_1 < r_4$,
3° $r_2 < r_3$, and
4° $r_3 + r_4 \geq 1$, what is
  (a) $r_3 + r_4 > 1$, or
  (b) $r_3 + r_4 = 1$

If any of the inequalities 1°–4° does not hold, any complete c.s. will contain a clash, as it is impossible to c.s.-satisfy simultaneously (i.e. in one and same c.s.) the constraints in which are the rationals, taking part in the failed inequality.

We can combine that four inequalities just in one taking into account the kind of interaction between part restrictions. $x\!:\!Wr_3R.C$ means that $\sim C$ has to be assigned to not greater than r_3 part of all R-successors of x, i.e. that C has to be assigned to at least $(1-r_3)$ part of them. We set $\check{r} = \max\big(\{r_1, 1-r_3\}\big)$ (or, if the constraint with one of r_1, r_3 is not in the cluster, $\check{r}$ is just the expression with the other). Now, it is obvious that if C is assigned to greater than $\check{r}$ part of all R-successors of x, then both $x\!:\!Mr_1R.C$ and $x\!:\!Wr_3R.C$ are (or the single one from the couple, which is in the cluster, is) c.s.-satisfied. Analogues reasonings go with the other couple of constraints, acting in one and the same direction (the ones with Mr_2 and Wr_4), and (a part smaller than) $\hat{r} = \min\big(\{1-r_2, r_4\}\big)$.

We call the constraints with part restrictions which determine $\check{r}$ and $\hat{r}$ the *dominating* constraints.

It is important to note that $r_3 + r_4 = 1$ does not spoil the c.s-satisfiability (unlike $r_1 + r_2 = 1$). We exclude this special sub-case from the general examination, and discuss it separately. Thus, case B. divides into two sub-cases:

(a) The cluster either contains no two W-constraints, or $r_3 + r_4 \neq 1$.
(b) The cluster contains the constraints with Wr_3 and Wr_4, and $r_3 + r_4 = 1$.

Clusters of type B(a) Our first claim is:

Lemma 1. *For a B(a)-cluster $Cl_x^S(R,C)$,*

$$\check{r} < \hat{r}$$

iff all inequalities from among 1°, 2°, 3°, and 4°(a), with the corresponding constraints being in the cluster, hold.

Proofs can be found in the Appendix.

Corollary 1. *$\check{r} < \hat{r}$ is a necessary condition for the c.s.-satisfiability of a B(a)-cluster $Cl_x^S(R, C)$.*

The upper inequality is also a *sufficient condition* for cluster's c.s.-satisfiability. Indeed, if $\check{r} < \hat{r}$, and the number of R-successors of x with C assigned—$|R^S(x, C)|$—is strongly (due to the strong inequality in the $\rightarrow_M$-rule) between $\check{r}.|R^S(x)|$ and $\hat{r}.|R^S(x)|$, then the dominating constraints are c.s.-satisfied, and so are the rest of the constraints in the cluster, if any. This shows that $\check{r} < \hat{r}$ ensures c.s.-satisfiability, but practical c.s.-satisfaction of a cluster is connected with the number of successors.

Note also that even though $\check{r} < \hat{r}$ holds, we can have *instable* c.s.-satisfaction, as can be seen from the next example. Let the dominating constraints be $x : M_{2/3}R.C$ and $x : M_{1/4}R.{\sim}C$. They can be S-satisfied if $R^S(x)$ has 10 variables (with C assigned to 7, and ${\sim}C$—to 3 of them), and also 11 variables (with the assignment $C : {\sim}C$—$8 : 3$), while if $R^S(x)$ has 12 variables, there is no way these constraints to be S-satisfied, as the first wants C to be assigned to at least 9, and the second—${\sim}C$ to at least 4 successors. In case of 13 R-successors of x the constraints again can be simultaneously c.s.-satisfied.

Definition 3. A cluster $Cl_x^S(R, C)$ is *n-satisfiable,* where $n \geq 0$, if it can be c.s.-satisfied when x has exactly n R-successors.

A cluster is *stably n-satisfiable,* if it is n-satisfiable, and for any natural number $n' > n$ it is also n'-satisfiable.

A cluster is *stably c.s.-satisfiable,* if it is stably n-satisfiable for some $n \geq 0$.

Note that from the above definition it follows that if a cluster is stably n-satisfiable, it is also stably n'-satisfiable, for any natural number $n' > n$.

In the example above the cluster is 10-, and 11-satisfiable, it is not 12-satisfiable, and it is (in fact—stably) 13-satisfiable.

So, if we have a sufficient condition for stable n-satisfiability of B(a)-clusters, we will know exactly when in the non-deterministic c.s. generation process stable c.s.-satisfaction of such a cluster will be surely achieved in at least one non-deterministic generation (we call it a *successful* generation). Then we will key at that moment the corresponding rules with respect to the constraints of that cluster, thus avoiding infinite rules application in the unsuccessful generations.

Lemma 2. *Let, for a B(a)-cluster $Cl_x^S(R, C)$, $\check{r} < \hat{r}$ hold. Then*

$$|R^S(x)| > \tfrac{1}{\hat{r} - \check{r}} \quad (\sharp)$$

is a sufficient condition for the non-deterministic $|R^S(x)|$-satisfiability of the cluster.

Lemma 3. *Let, for a B(a)-cluster $Cl_x^S(R, C)$, $\check{r} < \hat{r}$ and $(\sharp)$ hold, and the dominating constraints in the cluster be S-satisfied. Then any generating rule can always be applied in a way to yield S' such that the cluster is S'-satisfied.*

Lemma 3 shows that (♯) also guarantees the stability of the non-deterministic c.s.-satisfiability, namely stably $\left(\lfloor \frac{1}{\hat{r}-\check{r}} \rfloor + 1\right)$-satisfiability. It is clear that being once fulfilled, (♯) holds for any greater number of R-successors of x, and so c.s.-satisfying of the dominating constraints can be preserved as $R^{c.s.}(x)$ grows.

Thus, Lemma 2 and Lemma 3 guarantee for a c.s.-satisfiable (with $\check{r} < \hat{r}$) B(a)-cluster $Cl_x^S(R, C)$ that, having the number of R-successors of x equal to (or greater than) $\lfloor \frac{1}{\hat{r}-\check{r}} \rfloor + 1$ (what we will call the *border amount of successors, BAS,* for that cluster), the cluster can be non-deterministically c.s.-satisfied. Then, the termination of application of rules, triggered by (the constraints in) that cluster, is ensured by the check-up for $|R^S(x)|$.

Shortly said, any c.s.-satisfiable B(a)-cluster can be non-deterministically stably c.s.-satisfied when the current variable has enough many successors on the role in the cluster. We will rate that in the general case for all (possibly counteracting) constraints, to preserve from infinite application of rules.

Clusters of type B(b) B(b)-clusters are determined by the equality 4°(b) $r_3 + r_4 = 1$ for the rationals in W constructors. These clusters are c.s.-satisfiable if 2° $r_1 < r_4$, and 3° $r_2 < r_3$ hold (in case the corresponding M constructors are in the cluster). Thus, in case that 2° and 3° hold, or some of constraints with M constructors are missing, 4°(b) can be considered as a sufficient condition for the c.s.-satisfiability of a B(b)-cluster.

Lemma 4. *Let, for a B(b)-cluster $Cl_x^S(R, C)$, $r_1 < r_4$ and $r_2 < r_3$ hold, in case the corresponding M constructors are in the cluster. Then the cluster is c.s.-satisfiable, and the sufficient condition it to be non-deterministically c.s.-satisfied is the number of R-successors of x to be devisable by the denominator of r_3 and r_4 in the cluster.*

The general case Let us recall that the rules application strategy requires all possible applications of non-generating rules for the current variable to be done before the first application of a generating rule, what ensures all 'successor' constraints to be present in the c.s. at that time. Generalizing the considerations for counteracting in clusters, and using the cluster technique, we prove:

Lemma 5. *Let all possible applications of non-generating rules for the current variable x be done. Then it can be calculated a natural number $BAS_x \geq 1$, depending on the constraints at x, and having the following property: all part constraints at x which are simultaneously c.s.-satisfiable can be non-deterministically simultaneously c.s.-satisfied when the number of successors of x on any role in that constraints becomes equal to BAS_x.*

Lemma 5 both legitimates the use of BAS_x in the part rules applicability check-up, thus ensuring termination, and guarantees that all simultaneously c.s.-satisfiable part constraints will be non-deterministically simultaneously c.s.-satisfied, so that there would not be clashes with them in the complete c.s.

4 Correctness and complexity of the algorithm

Lemma 6. *The presented algorithm is a non-deterministic decision procedure for the satisfiability of an $\mathcal{ALCQPR}$-concept.*

Proof. (Sketch) We prove the correctness of the completion algorithm, proving the three properties of the c.s. generation process: termination, soundness, and completeness. As we have ensured both the termination of part rules application, and leaving no avoidable clashes with part constraints in the complete c.s., the proofs follow the scheme from [1] with no difficulties.

Lemma 7. *The completion algorithm can be implemented in PSPACE.*

Proof. (Sketch) We present such an implementation.

Then, the next theorem follows simply from the PSPACE hardness of $\mathcal{ALC}$, shown in [5], the above lemma, and Savitch's theorem.

Theorem 1 (PSPACE completeness). *Subsumption and satisfiability in $\mathcal{ALCQPR}$ are PSPACE-complete in the length of input independently of the notation of numbers.*

Open questions

It is known that in the language with counting constructors transitivity of relations brings high complexity, and, as it can be seen in [7], even undecidability. As a next step in the exploration of rational grading, a DL with only part restrictions as grading constructors, and transitive relations can be considered. Is the satisfiability for such DL decidable? And if so, how complex is it?

Acknowledgments

I would like to thank the anonymous referees for their valuable comments and suggestions.

References

1. Tobies, S.: PSPACE Reasoning for Graded Modal Logics. Journal of Logic and Computation 11(1), 85–106 (2001)
2. Pacuite, E., Salame, S.: Majority Logic. Principles of Knowledge Representation and Reasoning (KR 2004), 598–605 (2004)
3. Tinchev, T., Yanchev, M.: Modal Operators for Rational Grading. International Conference Pioneers of Bulgarian Mathematics, Book of Abstracts, 123-124 (2006)
4. Demri, S., Lugiez, D.: Presburger Modal Logic Is PSPACE-Complete. Automated Reasoning (IJCAR 2006) LNCS 4130, 541–556 (2006).
5. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)
6. Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W.: The Complexity of Concept Languages. Information and Computation 134(1), 1–58 (1997)
7. Horrocks, I., Sattler, U., Tobies, S.: Practical Reasoning for Expressive Description Logics. Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning (LPAR'99), 161-180 (1999)