PSPACE Reasoning for Graded Modal Logics*

Stephan Tobies
LuFg Theoretical Computer Science, RWTH Aachen,
Theoretische Informatik, Ahornstr. 55, D-52074 Aachen, Germany
tobies@informatik.rwth-aachen.de

Abstract

We present a PSPACE algorithm that decides satisfiability of the graded modal logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$—a natural extension of propositional modal logic $\mathbf{K}_{\mathcal{R}}$ by counting expressions—which plays an important role in the area of knowledge representation. The algorithm employs a tableaux approach and is the first known algorithm which meets the lower bound for the complexity of the problem. Thus, we exactly fix the complexity of the problem and refute a EXPTIME-hardness conjecture. We extend the results to the logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_{\cap}^{-1}})$, which augments $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ with inverse relations and intersection of accessibility relations. This establishes a kind of "theoretical benchmark" that all algorithmic approaches can be measured against.

Keywords: Modal Logic, Graded Modalities, Counting, Description Logic, Complexity.

1 Introduction

Propositional modal logics have found applications in many areas of computer science. Especially in the area of knowledge representation, the description logic (DL) $\mathcal{ALC}$, which is a syntactical variant of the propositional (multi-)modal logic $\mathbf{K}_{\mathcal{R}}$ [Sch91], forms the basis of a large number of formalisms used to represent and reason about conceptual and taxonomical knowledge of the application domain. The graded modal logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ extends $\mathbf{K}_{\mathcal{R}}$ by *graded modalities* [Fin72], i.e., counting expressions which allow one to express statements of the form "there are at least (at most) n accessible worlds that satisfy ... ". This is especially useful in knowledge representation because (a) humans tend to describe objects by the number of other objects they are related to (a stressed person is a person given at least three assignments that are urgent), and (b) qualifying number restrictions (the DL's analogue for graded modalities [HB91]) are necessary for modeling semantic data models [CLN94].

$\mathbf{K}_{\mathcal{R}}$ is decidable in PSPACE and can be embedded into a decidable fragment of predicate logic [AvBN98]. Hence, there are two general approaches for reasoning with $\mathbf{K}_{\mathcal{R}}$: dedicated decision procedures [Lad77, SSS91, GS96], and the translation into first order logic followed by the application of an existing first order theorem prover [OS97, Sch97]. To compete with the dedicated algorithms, the second approach has to yield a decision procedure and it has

*This papers appeared in the Journal of Logic and Computation, Vol. 10 No. 99-47, pp. 1–22 2000.

to be efficient, because the dedicated algorithms usually have optimal worst-case complexity. For $\mathbf{K}_{\mathcal{R}}$, the first issue is solved and, regarding the complexity, experimental results show that the algorithm competes well with dedicated algorithms [HS97]. Since experimental result can only be partially satisfactory, a theoretical complexity result would be desirable, but there are no exact results on the complexity of the theorem prover approach.

The situation for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ is more complicated: $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ is known to be decidable, but this result is rather recent [HB91], and the known PSPACE upper complexity bound for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ is only valid if we assume unary coding of numbers in the input, which is an unnatural restriction. For binary coding no upper bound is known and the problem has been conjectured to be EXPTIME-hard [dHR95]. This coincides with the observation that a straightforward adaptation of the translation technique leads to an exponential blow-up in the size of the first order formula. This is because it is possible to store the number n in $\log_k n$-bits if numbers are represented in k-ary coding. In [OSH96] a translation technique that overcomes this problem is proposed, but a decision procedure for the target fragment of first order logic yet has to be developed.

In this work we show that reasoning for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ is not harder than reasoning for $\mathbf{K}_{\mathcal{R}}$ by presenting an algorithm that decides satisfiability in PSPACE, even if the numbers in the input are binary coded. It is based on the tableaux algorithms for $\mathbf{K}_{\mathcal{R}}$ and tries to prove the satisfiability of a given formula by explicitly constructing a model for it. When trying to generalise the tableaux algorithms for $\mathbf{K}_{\mathcal{R}}$ to deal with $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$, there are some difficulties: (1) the straightforward approach leads to an incorrect algorithm; (2) even if this pitfall is avoided, special care has to be taken in order to obtain a space-efficient solution. As an example for (1), we will show that the algorithm presented in [dHR95] to decide satisfiability of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ is incorrect. Nevertheless, this algorithm will be the basis of our further considerations. Problem (2) is due to the fact that tableaux algorithms try to prove the satisfiability of a formula by explicitly building a model for it. If the tested formula requires the existence of n accessible worlds, a tableaux algorithm will include them in the model it constructs, which leads to exponential space consumption, at least if the numbers in the input are not unarily coded or memory is not re-used. An example for a correct algorithm which suffers from this problem can be found in [HB91] and is briefly presented in this paper. Our algorithm overcomes this problem by organising the search for a model in a way that allows for the re-use of space *for each successor*, thus being capable of deciding satisfiability of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ in PSPACE.

Using an extension of these techniques we obtain a PSPACE algorithm for the logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$, which extends $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ by inverse relations and intersection of relations. This solves an open problem from [DLNN97].

This paper is an significantly extended and improved version of [Tob99].

2 Preliminaries

In this section we introduce the graded modal logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$, the extension of the multimodal logic $\mathbf{K}_{\mathcal{R}}$ with graded modalities, first introduced in [Fin72].

DEFINITION 2.1 (SYNTAX AND SEMANTICS OF $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$)
Let $\mathcal{P} = \{p_0, p_1, \dots\}$ be a set of propositional atoms and $\mathcal{R}$ a set of *relation names*. The set of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$-*formulae* is built according to the following rules:

1. every propositional atom is a $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$-formula, and

2. if ϕ, ψ_1, ψ_2 are $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$-formulae, $n \in \mathbb{N}$, and R is a relation name, then $\neg\phi$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\langle R \rangle_n \phi$, and $[R]_n \phi$ are formulae.

The semantics of $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$-formulae is based on *Kripke structures*

$$\mathfrak{M} = (W^\mathfrak{M}, \{R^\mathfrak{M} \mid R \in \mathcal{R}\}, V^\mathfrak{M}),$$

where $W^\mathfrak{M}$ is a non-empty set of worlds, each $R^\mathfrak{M} \subseteq W^\mathfrak{M} \times W^\mathfrak{M}$ is an *accessibility relation* on worlds (for $R \in \mathcal{R}$), and $V^\mathfrak{M}$ is a *valuation* assigning subsets of $W^\mathfrak{M}$ to the propositional atoms in $\mathcal{P}$. For a Kripke structure $\mathfrak{M}$, an element $x \in W^\mathfrak{M}$, and a $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$-formula, the model relation $\models$ is defined inductively on the structure of formulae:

$$\begin{aligned}
&\mathfrak{M}, x \models p \text{ iff } x \in V^\mathfrak{M}(p) \text{ for } p \in \mathcal{P} \\
&\mathfrak{M}, x \models \neg\phi \text{ iff } \mathfrak{M}, x \not\models \phi \\
&\mathfrak{M}, x \models \psi_1 \wedge \psi_2 \text{ iff } \mathfrak{M}, x \models \psi_1 \text{ and } \mathfrak{M}, x \models \psi_2 \\
&\mathfrak{M}, x \models \psi_1 \vee \psi_2 \text{ iff } \mathfrak{M}, x \models \psi_1 \text{ or } \mathfrak{M}, x \models \psi_2 \\
&\mathfrak{M}, x \models \langle R \rangle_n \phi \text{ iff } \sharp R^\mathfrak{M}(x, \phi) > n \\
&\mathfrak{M}, x \models [R]_n \phi \text{ iff } \sharp R^\mathfrak{M}(x, \neg\phi) \leq n
\end{aligned}$$

where $\sharp R^\mathfrak{M}(x, \phi) := |\{y \in W^\mathfrak{M} \mid (x, y) \in R^\mathfrak{M} \text{ and } \mathfrak{M}, y \models \phi\}|$

The propositional modal logic $\mathbf{K}_\mathcal{R}$ is defined as the fragment of $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$ in which for all modal operators $n = 0$ holds.

A formula is called *satisfiable* iff there exists a structure $\mathfrak{M}$ and a world $x \in W^\mathfrak{M}$ such that $\mathfrak{M}, x \models \phi$.

By $\mathrm{SAT}(\mathbf{K}_\mathcal{R})$and $\mathrm{SAT}(\mathbf{Gr}(\mathbf{K}_\mathcal{R}))$we denote the sets of satisfiable formulae of $\mathbf{K}_\mathcal{R}$ and $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$, respectively.

As usual, the modal operators $\langle R \rangle_n$ and $[R]_n$ are dual: $\sharp R^\mathfrak{M}(x, \phi) > n$ means that in $\mathfrak{M}$ more than n R-successors of x satisfy ϕ; $\sharp R^\mathfrak{M}(x, \neg\phi) \leq n$ means that in $\mathfrak{M}$ all but at most n R-successors satisfy ϕ.

In the following we will only consider formulae in *negation normal form* (NNF), a form in which negations have been pushed inwards and occur in front of propositional atoms only. We will denote the NNF of $\neg\phi$ by $\sim\phi$. The NNF can always be generated in linear time and space by successively applying the following equivalences from left to right:

$$\begin{aligned}
\neg(\psi_1 \wedge \psi_2) &\equiv \neg\psi_1 \vee \neg\psi_2 \qquad & \neg\langle R \rangle_n \psi &\equiv [R]_n \neg\psi \\
\neg(\psi_1 \vee \psi_2) &\equiv \neg\psi_1 \wedge \neg\psi_2 \qquad & \neg[R]_n \psi &\equiv \langle R \rangle_n \neg\psi
\end{aligned}$$

3 Reasoning for $\mathrm{Gr}(\mathrm{K}_\mathcal{R})$

Before we present our algorithm for deciding satisfiability of $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$, for historic and didactic reasons, we present two other solutions: an incorrect one [dHR95], and a solution that is less efficient [HB91].

From the fact that $\mathrm{SAT}(\mathbf{K}_\mathcal{R})$ is PSPACE-complete [Lad77, HM92], it immediately follows, that $\mathrm{SAT}(\mathbf{Gr}(\mathbf{K}_\mathcal{R}))$ is PSPACE-hard. The algorithms we will consider decide the satisfiability of a given formula ϕ by trying to construct a model for ϕ.

3.1 An incorrect algorithm

In [dHR95], an algorithm for deciding $\mathrm{SAT}(\mathbf{Gr}(\mathbf{K}_{\mathcal{R}}))$ is given, which, unfortunately, is incorrect. Nevertheless, it will be the basis for our further considerations and thus it is presented here. It will be referred to as the *incorrect* algorithm. It is based on an algorithm given in [DLNN97] to decide the satisfiability of the DL $\mathcal{ALCNR}$, which basically is the restriction of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$, where, in formulae of the form $\langle R\rangle_n\phi$ or $[R]_n\phi$ with $n > 0$, necessarily $\phi = p \vee \neg p$ holds.

The algorithm for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ tries to build a model for a formula ϕ by manipulating sets of constraints with the help of so-called *completion rules*. This is a well-known technique to check the satisfiability of modal formulae, which has already been used to prove decidability and complexity results for other DLs (e. g., [SSS91, HB91, BBH96]). These algorithms can be understood as variants of tableaux algorithms which are used, for example, to decide satisfiability of the modal logics $\mathbf{K}_{\mathcal{R}}$, $\mathbf{T}_{\mathcal{R}}$, or $\mathbf{S4}_{\mathcal{R}}$ in [HM92].

DEFINITION 3.1

Let $\mathcal{V}$ be a set of variables. A *constraint system* (c.s.) S is a finite set of expressions of the form '$x \models \phi$' and 'Rxy', where ϕ is a formula, $R \in \mathcal{R}$, and $x, y \in \mathcal{V}$.

For a c.s. S, let $\sharp R^S(x,\phi)$ be the number of variables y for which $\{Rxy, y \models \phi\} \subseteq S$. The c.s. $[z/y]S$ is obtained from S by replacing every occurrence of y by z; this replacement is said to be *safe* iff, for every variable x, formula ϕ, and relation symbol R with $\{x \models \langle R\rangle_n\phi, Rxy, Rxz\} \subseteq S$ we have $\sharp R^{[z/y]S}(x,\phi) > n$.

A c.s. S is said to contain a *clash*, iff for a propositional atom p, a formula ϕ, and $m \leq n$:

$$\{x \models p, x \models \neg p\} \subseteq S \text{ or } \{x \models \langle R\rangle_m\phi, x \models [R]_n{\sim}\phi\} \subseteq S.$$

Otherwise it is called *clash-free*. A c.s. S is called *complete* iff none of the rules given in Fig. 1 is applicable to S.

To test the satisfiability of a formula ϕ, the incorrect algorithm works as follows: it starts with the c.s. $\{x \models \phi\}$ and successively applies the rules given in Fig. 1, stopping if a clash is occurs. Both the rule to apply and the formula to add (in the $\rightarrow_\vee$-rule) or the variables to identify (in the $\rightarrow_\leq$-rule) are selected non-deterministically. The algorithm answers "ϕ is satisfiable" iff the rules can be applied in a way that yields a complete and clash-free c.s. The notion of *safe* replacement of variables is needed to ensure the termination of the rule application [HB91].

Since we are interested in PSPACE algorithms, non-determinism imposes no problem due to Savitch's Theorem, which states that deterministic and non-deterministic polynomial space coincide [Sav70].

To prove the correctness of a non-deterministic completion algorithm, it is sufficient to prove three properties of the model generation process:

1. Termination: Any sequence of rule applications is finite.
2. Soundness: If the algorithm terminates with a complete and clash-free c.s. S, then the tested formula is satisfiable.
3. Completeness: If the formula is satisfiable, then there is a sequence of rule applications that yields a complete and clash-free c.s.

$\rightarrow_\wedge$-rule: if 1. $x \models \psi_1 \wedge \psi_2 \in S$ and
2. $\{x \models \psi_1, x \models \psi_2\} \not\subseteq S$
then $S \rightarrow_\wedge S \cup \{x \models \psi_1, x \models \psi_2\}$

$\rightarrow_\vee$-rule: if 1. $(x \models \psi_1 \vee \psi_2) \in S$ and
2. $\{x \models \psi_1, x \models \psi_2\} \cap S = \emptyset$
then $S \rightarrow_\vee S \cup \{x \models \chi\}$ where $\chi \in \{\psi_1, \psi_2\}$

$\rightarrow_>$-rule: if 1. $x \models \langle R \rangle_n \phi \in S$ and
2. $\sharp R^S(x, \phi) \leq n$
then $S \rightarrow_> S \cup \{Rxy, y \models \phi\}$ where y is a fresh variable.

$\rightarrow_{\leq 0}$-rule: if 1. $x \models [R]_0 \phi, Rxy \in S$ and
2. $y \models \phi \notin S$
then $S \rightarrow_{\leq 0} S \cup \{y \models \phi\}$

$\rightarrow_\leq$-rule: if 1. $x \models [R]_n \phi, \sharp R^S(x, \phi) > n > 0$ and
2. $Rxy, Rxz \in S$ and
3. replacing y by z is safe in S
then $S \rightarrow_\leq [z/y]S$

Figure 1: The incorrect completion rules for $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$.

The error of the incorrect algorithm is, that is does not satisfy Property 2, even though the converse is claimed:

> CLAIM([dHR95]): Let ϕ be a $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$-formula in NNF. ϕ is satisfiable iff $\{x_0 \models \phi\}$ can be transformed into a clash-free complete c.s. using the rules from Figure 1.

Unfortunately, the *if*-direction of this claim is not true, which we will prove by a simple counterexample. Consider the formula

$$\phi = \langle R \rangle_2 p_1 \wedge [R]_1 p_2 \wedge [R]_1 \neg p_2.$$

On the one hand, ϕ is not satisfiable. Assume $\mathfrak{M}, x \models \langle R \rangle_2 p_1$. This implies the existence of at least three R-successors y_1, y_2, y_3 of x. For each of the y_i either $\mathfrak{M}, y_i \models p_2$ or $\mathfrak{M}, y_i \not\models p_2$ holds by the definition of $\models$. Without loss of generality, there are two worlds y_{i_1}, y_{i_2} such that $\mathfrak{M}, y_{i_j} \models p_2$, which implies $\mathfrak{M}, x \not\models [R]_1 \neg p_2$ and hence $\mathfrak{M}, x \not\models \phi$.

On the other hand, the c.s. $S = \{x \models \phi\}$ can be turned into a complete and clash-free c.s. using the rules from Fig. 1, as is shown in Fig. 2. Clearly this invalidates the claim and its proof.

3.2 An alternative syntax

At this stage the reader may have noticed the cumbersome semantics of the $[R]_n$ operator, which origins from the wish that the duality $\Box\phi \equiv \neg\Diamond\neg\phi$ of $\mathbf{K}$ carries over to $[R]_n \phi \equiv \neg\langle R \rangle_n \neg\phi$ in $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$. This makes the semantics of $[R]_n$ and $\langle R \rangle_n$ un-intuitive. Not only does the n in a diamond operator mean "more than n" while it means "less *or equal* than n" for a box operator. The semantics also introduce a "hidden" negation.

$$\{x \models \phi\} \rightarrow_{\wedge} \cdots \rightarrow_{\wedge} \underbrace{\{x \models \phi,\ x \models \langle R\rangle_2 p_1,\ x \models [R]_1 p_2,\ x \models [R]_1 \neg p_2\}}_{=S_1}$$

$$\rightarrow_{>} \cdots \rightarrow_{>} \underbrace{S_1 \cup \{Rxy_i,\ y_i \models p_1 \mid i = 1,2,3\}}_{=S_2}$$

S_2 is clash-free and complete, because $\sharp R^{S_2}(x, p_1) = 3$ and $\sharp R^{S_2}(x, p_2) = 0$.

Figure 2: A run of the incorrect algorithm.

To overcome these problems, we will replace these modal operators by a syntax inspired by the counting quantifiers in predicate logic: the operators $\langle R\rangle_{\leq n}$ and $\langle R\rangle_{\geq n}$ with semantics defined by :

$$\mathfrak{M}, x \models \langle R\rangle_{\leq n}\phi \text{ iff } \sharp R^{\mathfrak{M}}(x,\phi) \leq n,$$
$$\mathfrak{M}, x \models \langle R\rangle_{\geq n}\phi \text{ iff } \sharp R^{\mathfrak{M}}(x,\phi) \geq n.$$

This modification does not change the expressivity of the language, since $\mathfrak{M}, x \models \langle R\rangle_n\phi$ iff $\mathfrak{M}, x \models \langle R\rangle_{\geq n+1}\phi$ and $\mathfrak{M}, x \models [R]_n\phi$ iff $\mathfrak{M}, x \models \langle R\rangle_{\leq n}\neg\phi$. We use the following equivalences to transform formulae in the new syntax into NNF:

$$\neg\langle R\rangle_{\geq 0}\phi \equiv p \wedge \neg p$$
$$\neg\langle R\rangle_{\geq n}\phi \equiv \langle R\rangle_{\leq n-1}\phi \text{ iff } n > 1$$
$$\neg\langle R\rangle_{\leq n}\phi \equiv \langle R\rangle_{\geq n+1}\phi$$

3.3 A correct but inefficient solution

To understand the mistake of the incorrect algorithm, it is useful to know how soundness is usually established for the kind of algorithms we consider. The underlying idea is that a complete and clash-free c.s. induces a model for the formula tested for satisfiability:

DEFINITION 3.2 (CANONICAL STRUCTURE)
Let S be a c.s. The *canonical structure* $\mathfrak{M}_S = (W^{\mathfrak{M}_S}, \{R^{\mathfrak{M}_S} \mid R \in \mathcal{R}\}, V^{\mathfrak{M}_S})$ *induced by* S is defined as follows:

$$W^{\mathfrak{M}_S} = \{x \in \mathcal{V} \mid x \text{ occurs in } S\},$$
$$R^{\mathfrak{M}_S} = \{(x,y) \in \mathcal{V}^2 \mid Rxy \in S\},$$
$$V^{\mathfrak{M}_S}(p) = \{x \in \mathcal{V} \mid x \models p \in S\}.$$

Using this definition, it is then easy to prove that the canonical structure induced by a complete and clash-free c.s. is a model for the tested formula.

The mistake of the incorrect algorithm is due to the fact that it did not take into account that, in the canonical model induced by a complete and clash-free c.s., there are formulae satisfied by the worlds even though these formulae do not appear as constraints in the c.s. Already in [HB91], an algorithm very similar to the incorrect one is presented which decides the satisfiability of $\mathcal{ALCQ}$, a notational variant of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$.

$\rightarrow_\wedge$-, $\rightarrow_\vee$-rule:	see Fig. 1
$\rightarrow_{\text{choose}}$-rule:	if 1. $x \models \langle R \rangle_{\bowtie n} \phi, Rxy \in S$ and 2. $\{y \models \phi, y \models \sim\phi\} \cap S = \emptyset$ then $S \rightarrow_{\text{choose}} S \cup \{y \models \chi\}$ where $\chi \in \{\phi, \sim\phi\}$
$\rightarrow_\geq$-rule:	if 1. $x \models \langle R \rangle_{\geq n} \phi \in S$ and 2. $\sharp R^S(x, \phi) < n$ then $S \rightarrow_\geq S \cup \{Rxy, y \models \phi\}$ where y is a new variable.
$\rightarrow_\leq$-rule:	if 1. $x \models \langle R \rangle_{\leq n} \phi, \sharp R^S(x, \phi) > n$ and 2. $y \neq z, Rxy, Rxz, y \models \phi, z \models \phi \in S$ and 3. the replacement of y by z is safe in S then $S \rightarrow_\leq [y/z]S$

Figure 3: The standard completion rules

The algorithm essentially uses the same definitions and rules. The only differences are the introduction of the $\rightarrow_{\text{choose}}$-rule and an adaption of the $\rightarrow_\geq$-rule to the alternative syntax. The $\rightarrow_{\text{choose}}$-rule makes sure that all "relevant" formulae that are implicitly satisfied by a variable are made explicit in the c.s. Here, relevant formulae for a variable y are those occuring in modal formulae in constraints for variables x such that Rxy appears in the c.s. The complete rule set for the modified syntax of $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$ is given in Fig. 3. The definition of *clash* has to be modified as well: A c.s. S contains a clash iff

- $\{x \models p, x \models \neg p\} \subseteq S$ for some variable x and a propositional atom p, or
- $x \models \langle R \rangle_{\leq n} \phi \in S$ and $\sharp R^S(x, \phi) > n$ for some variable x, relation R, formula ϕ, and $n \in \mathbb{N}$.

Furthermore, the notion of safe replacement has to be adapted to the new syntax: the replacement of y by z in S is called *safe* iff, for every variable x, formula ϕ, and relation symbol R with $\{x \models \langle R \rangle_{\geq n} \phi, Rxy, Rxz\} \subseteq S$ we have $\sharp R^{[z/y]S}(x, \phi) \geq n$.

The algorithm, which works like the incorrect algorithm but uses the expansion rules from Fig. 3—where $\bowtie$ is used as a placeholder for either $\leq$ or $\geq$—and the definition of clash from above will be called the *standard algorithm*; it is a decision procedure for SAT($\mathbf{Gr}(\mathbf{K}_\mathcal{R})$):

THEOREM 3.3 ([HB91])
Let ϕ be a $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$-formula in NNF. ϕ is satisfiable iff $\{x_0 \models \phi\}$ can be transformed into a clash-free complete c.s. using the rules in Figure 3. Moreover, each sequence of these rule-applications is finite.

While no complexity result is explicitly given in [HB91], it is easy to see that a PSPACE result could be derived from the algorithm using the trace technique, employed in [SSS91] to show that satisfiability of $\mathcal{ALC}$, the notational variant for $\mathbf{K}_\mathcal{R}$, is decidable in PSPACE.

Unfortunately this is only true if we assume the numbers in the input to be unary coded. The reason for this lies in the $\rightarrow_\geq$-rule, which generates n successors for a formula of the form $\langle R \rangle_{\geq n} \phi$. If n is unary coded, these successors consume at least polynomial space in the size of the input formula. If we assume binary (or k-ary with $k > 1$) encoding, the space consumption is exponential in the size of the input because a number n can be represented in $\log_k n$ bits in k-ary coding. This blow-up can not be avoided because the completeness of the standard algorithm relies on the generation *and identification* of these successors, which makes it necessary to keep them in memory *at one time*.

4 An optimal solution

In the following, we will present the algorithm which will be used to prove the following theorem; it contradicts the EXPTIME-hardness conjecture in [dHR95].

THEOREM 4.1
Satisfiability for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ is PSPACE-complete if numbers in the input are represented using **binary** coding.

When aiming for a PSPACE algorithm, it is impossible to generate all successors of a variable in a c.s. at a given stage because this may consume space that is exponential in the size of the input concept. We will give an optimised rule set for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$-satisfiability that does not rely on the identification of successors. Instead we will make stronger use of non-determinism to guess the assignment of the relevant formulae to the successors by the time of their generation. This will make it possible to generate the c.s. in a depth first manner, which will facilitate the re-use of space.

The new set of rules is shown in Fig. 4. The algorithm that uses these rules is called the *optimised algorithm*. The definition of *clash* is taken from the standard algorithm. We do not need a $\rightarrow_{\leq}$-rule.

At first glance, the $\rightarrow_{\geq}$-rule may appear to be complicated and therefor is explained in more detail: like the standard $\rightarrow_{\geq}$-rule, it is applicable to a c.s. that contains the constraint $x \models \langle R \rangle_{\geq n} \phi$ if there are less than n R-successors y of x with $y \models \phi \in S$. The rule then adds a new successor y to S. Unlike the standard algorithm, the optimised algorithm also adds additional constraints of the form $y \models (\sim)\psi$ to S for each formula ψ appearing in a constraint of the form $x \models \langle R \rangle_{\bowtie n} \psi$. Since we have suspended the application of the $\rightarrow_{\geq}$-rule until no other rule applies to x, by this time S contains all constraints of the form $x \models \langle R \rangle_{\bowtie n} \psi$ it will ever contain. This combines the effects of both the $\rightarrow_{\text{choose}}$- and the $\rightarrow_{\leq}$-rule of the standard algorithm.

$\rightarrow_{\wedge}$-, $\rightarrow_{\vee}$-rule: see Fig. 1

$\rightarrow_{\geq}$-rule: if 1. $x \models \langle R \rangle_{\geq n} \phi \in S$, and
2. $\sharp R^S(x, \phi) < n$, and
3. neither the $\rightarrow_{\wedge}$- nor the $\rightarrow_{\vee}$-rule apply to a constraint for x
then $S \rightarrow_{\geq} S \cup \{Rxy, y \models \phi, y \models \chi_1, \ldots, y \models \chi_k\}$ where
$\{\psi_1, \ldots, \psi_k\} = \{\psi \mid x \models \langle R \rangle_{\bowtie m} \psi \in S\}$, $\chi_i \in \{\psi_i, \sim\psi_i\}$, and
y is a fresh variable.

Figure 4: The optimised completion rules.

4.1 Correctness of the optimised algorithm

To establish the correctness of the optimised algorithm, we will show its termination, soundness, and completeness.

To analyse the memory usage of the algorithm it is very helpful to view a c.s. as a graph: A c.s. S induces a labeled graph $G(S) = (N, E, \mathcal{L})$ with

- The set of nodes N is the set of variables appearing in S.
- The edges E are defined by $E := \{xy \mid Rxy \in S \text{ for some } R \in \mathcal{R}\}$.
- $\mathcal{L}$ labels nodes and edges in the following way:
  - For a node $x \in N$: $\mathcal{L}(x) := \{\phi \mid x \models \phi \in S\}$.
  - For an edge $xy \in E$: $\mathcal{L}(xy) := \{R \mid Rxy \in S\}$.

It is easy to show that the graph $G(S)$ for a c.s. S generated by the optimised algorithm from an initial c.s. $\{x_0 \models \phi\}$ is a tree with root x_0, and for each edge $xy \in E$, the label $\mathcal{L}(xy)$ is a singleton. Moreover, for each $x \in N$ it holds that $\mathcal{L}(x) \subseteq \mathit{clos}(\phi)$ where $\mathit{clos}(\phi)$ is the smallest set of formulae satisfying

- $\phi \in \mathit{clos}(\phi)$,
- if $\psi_1 \vee \psi_2$ or $\psi_1 \wedge \psi_2 \in \mathit{clos}(\phi)$, then also $\psi_1, \psi_2 \in \mathit{clos}(\phi)$,
- if $\langle R \rangle_{\bowtie n} \psi \in \mathit{clos}(\phi)$, then also $\psi \in \mathit{clos}(\phi)$,
- if $\psi \in \mathit{clos}(\phi)$, then also $\sim\psi \in \mathit{clos}(\phi)$.

We will use the fact that the number of elements of $\mathit{clos}(\phi)$ is bounded by $2 \times |\phi|$ where $|\phi|$ denotes the length of ϕ. This is easily shown by proving

$$\mathit{clos}(\phi) = \mathit{sub}(\phi) \cup \{\sim\psi \mid \psi \in \mathit{sub}(\phi)\}$$

where $\mathit{sub}(\phi)$ denotes the set of all sub-formulae of ϕ. The size of $\mathit{sub}(\phi)$ is obviously bounded by $|\phi|$.

4.1.1 Termination

First, we will show that the optimised algorithm always terminates, i.e., each sequence of rule applications starting from a c.s. of the form $\{x_0 \models \phi\}$ is finite. The next lemma will also be of use when we will consider the complexity of the algorithm.

LEMMA 4.2
Let ϕ be a formula in NNF and S a c.s. that is generated by the optimised algorithm starting from $\{x_0 \models \phi\}$.

- The length of a path in $G(S)$ is limited by $|\phi|$.
- The out-degree of $G(S)$ is bounded by $|\mathit{clos}(\phi)| \times 2^{|\phi|}$.

PROOF. For a variable $x \in N$, we define $\ell(x)$ as the maximum depth of nested modal operators in $\mathcal{L}(x)$. Obviously, $\ell(x_0) \leq |\phi|$ holds. Also, if $xy \in E$ then $\ell(x) > \ell(y)$. Hence each path $x_1, \ldots, x_k$ in $G(S)$ induces a sequence $\ell(x_1) > \cdots > \ell(x_k)$ of natural numbers. $G(S)$ is a tree with root x_0, hence the longest path in $G(S)$ starts with x_0 and its length is bounded by $|\phi|$.

Successors in $G(S)$ are only generated by the $\rightarrow_{\geq}$-rule. For a variable x this rule will generate at most n successors for each $\langle R \rangle_{\geq n}\psi \in \mathcal{L}(x)$. There are at most $|clos(\phi)|$ such formulae in $\mathcal{L}(x)$. Hence the out-degree of x is bounded by $|clos(\phi)| \times 2^{|\phi|}$, where $2^{|\phi|}$ is a limit for the biggest number that may appear in ϕ if binary coding is used. ∎

COROLLARY 4.3 (TERMINATION)
Any sequence of rule applications starting from a c.s. $S = \{x_0 \models \phi\}$ of the optimised algorithm is finite.

PROOF. The sequence of rules induces a sequence of trees. The depth and the out-degree of these trees is bounded in $|\phi|$ by Lemma 4.2. For each variable x the label $\mathcal{L}(x)$ is a subset of the finite set $clos(\phi)$. Each application of a rule either

- adds a constraint of the form $x \models \psi$ and hence adds an element to $\mathcal{L}(x)$, or
- adds fresh variables to S and hence adds additional nodes to the tree $G(S)$.

Since constraints are never deleted and variables are never identified, an infinite sequence of rule application must either lead to an arbitrary large number of nodes in the trees which contradicts their boundedness, or it leads to an infinite label of one of the nodes x which contradicts $\mathcal{L}(x) \subseteq clos(\phi)$. ∎

4.1.2 Soundness and Completeness

The following definition will be very helpful to establish soundness and completeness of the optimised algorithm:

DEFINITION 4.4
A c.s. S is called *satisfiable* iff there exists a Kripke structure $\mathfrak{M} = (W^{\mathfrak{M}}, \{R^{\mathfrak{M}} \mid R \in \mathcal{R}\}, V^{\mathfrak{M}})$ and a mapping $\alpha : \mathcal{V} \rightarrow W^{\mathfrak{M}}$ such that the following properties hold:

1. If y, z are distinct variables such that $Rxy, Rxz \in S$, then $\alpha(y) \neq \alpha(z)$.
2. If $x \models \psi \in S$ then $\mathfrak{M}, \alpha(x) \models \psi$.
3. If $Rxy \in S$ then $(\alpha(x), \alpha(y)) \in R^{\mathfrak{M}}$.

In this case, $\mathfrak{M}, \alpha$ is called a *model* of S.

It easily follows from this definition, that a c.s. S that contains a clash can not be satisfiable and that the c.s. $\{x_0 \models \phi\}$ is satisfiable if and only if ϕ is satisfiable.

LEMMA 4.5 (LOCAL CORRECTNESS)
Let S, S' be c.s. generated by the optimised algorithm from a c.s. of the form $\{x_0 \models \phi\}$.

1. If S' is obtained from S by application of the (deterministic) $\rightarrow_\wedge$-rule, then S is satisfiable if and only if S' is satisfiable.

2. If S' is obtained from S by application of the (non-deterministic) $\rightarrow_\vee$- or $\rightarrow_\geq$-rule, then S is satisfiable if S' is satisfiable. Moreover, if S is satisfiable, then the rule can always be applied in such a way that it yields a c.s. S' that is satisfiable.

PROOF. $S \rightarrow S'$ for any rule $\rightarrow$ implies $S \subseteq S'$, hence each model of S' is also a model of S. Consequently, we must show only the other direction.

1. Let $\mathfrak{M}, \alpha$ be a model of S and let $x \models \psi_1 \wedge \psi_2$ be the constraint that triggers the application of the $\rightarrow_\wedge$-rule. The constraint $x \models \psi_1 \wedge \psi_2 \in S$ implies $\mathfrak{M}, \alpha(x) \models \psi_1 \wedge \psi_2$. This implies $\mathfrak{M}, \alpha(x) \models \psi_i$ for $i = 1, 2$. Hence $\mathfrak{M}, \alpha$ is also a model of $S' = S \cup \{x \models \psi_1, x \models \psi_2\}$.

2. Firstly, we consider the $\rightarrow_\vee$-rule. Let $\mathfrak{M}, \alpha$ be a model of S and let $x \models \psi_1 \vee \psi_2$ be the constraint that triggers the application of the $\rightarrow_\vee$-rule. $x \models \psi_1 \vee \psi_2 \in S$ implies $\mathfrak{M}, \alpha(x) \models \psi_1 \vee \psi_2$. This implies $\mathfrak{M}, \alpha(x) \models \psi_1$ or $\mathfrak{M}, \alpha(x) \models \psi_2$. Without loss of generality we may assume $\mathfrak{M}, \alpha(x) \models \psi_1$. The $\rightarrow_\vee$-rule may choose $\chi = \psi_1$, which implies $S' = S \cup \{x \models \psi_1\}$ and hence $\mathfrak{M}, \alpha$ is a model for S'.

   Secondly, we consider the $\rightarrow_\geq$-rule. Again let $\mathfrak{M}, \alpha$ be a model of S and let $x \models \langle R \rangle_{\geq n} \phi$ be the constraint that triggers the application of the $\rightarrow_\geq$-rule. Since the $\rightarrow_\geq$-rule is applicable, we have $\sharp R^S(x, \phi) < n$. We claim that there is a $w \in W^{\mathfrak{M}}$ with

$$(\alpha(x), w) \in R^{\mathfrak{M}}, \mathfrak{M}, w \models \phi, \text{ and } w \notin \{\alpha(y) \mid Rxy \in S\}. \qquad (*)$$

   Before we prove this claim, we show how it can be used to finish the proof. The world w is used to "select" a choice of the $\rightarrow_\geq$-rule that preserves satisfiability: Let $\{\psi_1, \ldots, \psi_n\}$ be an enumeration of the set $\{\psi \mid x \models \langle R \rangle_{\bowtie n} \psi \in S\}$. We set

$$S' = S \cup \{Rxy, y \models \phi\} \cup \{y \models \psi_i \mid \mathfrak{M}, w \models \psi_i\} \cup \{y \models \sim\psi_i \mid \mathfrak{M}, w \not\models \psi_i\}.$$

   Obviously, $\mathfrak{M}, \alpha[y \mapsto w]$ is a model for S' (since y is a fresh variable and w satisfies $(*)$), and S' is a possible result of the application of the $\rightarrow_\geq$-rule to S.

We will now come back to the claim. It is obvious that there is a w with $(\alpha(x), w) \in R^{\mathfrak{M}}$ and $\mathfrak{M}, w \models \phi$ that is not contained in $\{\alpha(y) \mid Rxy, y \models \phi \in S\}$, because $\sharp R^{\mathfrak{M}}(x, \phi) \geq n > \sharp R^S(x, \phi)$. Yet w might appear as the image of an element y' such that $Rxy' \in S$ but $y' \models \phi \notin S$.

Now, $Rxy' \in S$ and $y' \models \phi \notin S$ implies $y' \models \sim\phi \in S$. This is due to the fact that the constraint Rxy' must have been generated by an application of the $\rightarrow_\geq$-rule because it has not been an element of the initial c.s. The application of this rule was suspended until neither the $\rightarrow_\wedge$- nor the $\rightarrow_\vee$-rule are applicable to x. Hence, if $x \models \langle R \rangle_{\geq n} \phi$ is an element of S now, then it has already been in S when the $\rightarrow_\geq$-rule that generated y' was applied. The $\rightarrow_\geq$-rule guarantees that either $y' \models \phi$ or $y' \models \sim\phi$ is added to S. Hence $y' \models \sim\phi \in S$. This is a contradiction to $\alpha(y') = w$ because under the assumption that $\mathfrak{M}, \alpha$ is a model of S this would imply $\mathfrak{M}, w \models \sim\phi$ while we initially assumed $\mathfrak{M}, w \models \phi$. ∎

From the local completeness of the algorithm we can immediately derive the global completeness of the algorithm:

LEMMA 4.6 (COMPLETENESS)
If $\phi \in \text{SAT}(\mathbf{Gr}(\mathbf{K}_\mathcal{R}))$ in NNF, then there is a sequence of applications of the optimised rules starting with $S = \{x_0 \models \phi\}$ that results in a complete and clash-free c.s.

PROOF. The satisfiability of ϕ implies that also $\{x_0 \models \phi\}$ is satisfiable. By Lemma 4.5 there is a sequence of applications of the optimised rules which preserves the satisfiability of the c.s. By Lemma 4.3 any sequence of applications must be finite. No generated c.s. (including the last one) may contain a clash because this would make it unsatisfiable. ■

Note that since we have made no assumption about the order in which the rules are applied (with the exception that is stated in the conditions of the $\rightarrow_\geq$-rule), the selection of the constraints to apply a rule to as well as the selection which rule to apply is "don't-care" non-deterministic, i.e., if a formula is satisfiable, then this can be proved by an arbitrary sequence of rule applications. Without this property, the resulting algorithm certainly would be useless for practical applications, because any deterministic implementation would have to use backtracking for the selection of constraints and rules.

LEMMA 4.7 (SOUNDNESS)
Let ϕ be a $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$-formula in NNF. If there is a sequence of applications of the optimised rules starting with the c.s. $\{x_0 \models \phi\}$ that results in a complete and clash-free c.s., then $\phi \in \text{SAT}(\mathbf{Gr}(\mathbf{K}_\mathcal{R}))$.

PROOF. Let S be a complete and clash-free c.s. generated by applications of the optimised rules. We will show that the canonical model $\mathfrak{M}_S$ together with the identity function is a model for S. Since S was generated from $\{x_0 \models \phi\}$ and the rules do not remove constraints from the c.s., $x_0 \models \phi \in S$. Thus $\mathfrak{M}_S$ is also a model for ϕ with $\mathfrak{M}_S, x_0 \models \phi$.

By construction of $\mathfrak{M}_S$, Property 1 and 3 of Definition 4.4 are trivially satisfied. It remains to show that $x \models \psi \in S$ implies $\mathfrak{M}_S, x \models \psi$, which we will show by induction on the norm $\|\cdot\|$ of a formula ψ. The norm $\|\psi\|$ for formulae in NNF is inductively defined by:

$$
\begin{array}{lllll}
\|p\| & := & \|\neg p\| & := & 0 \quad \text{for } p \in \mathcal{P} \\
\|\psi_1 \wedge \psi_2\| & := & \|\psi_1 \vee \psi_2\| & := & 1 + \|\psi_1\| + \|\psi_2\| \\
\|\langle R \rangle_{\bowtie n} \psi\| & & & := & 1 + \|\psi\|
\end{array}
$$

This definition is chosen such that it satisfies $\|\psi\| = \|{\sim}\psi\|$ for every formula ψ.

- The first base case is $\psi = p$ for $p \in \mathcal{P}$. $x \models p \in S$ implies $x \in V^{\mathfrak{M}_S}(p)$ and hence $\mathfrak{M}_S, x \models p$. The second base case is $x \models \neg p \in S$. Since S is clash-free, this implies $x \models p \notin S$ and hence $x \notin V^{\mathfrak{M}_S}(p)$. This implies $\mathfrak{M}_S, x \models \neg p$.

- $x \models \psi_1 \wedge \psi_2 \in S$ implies $x \models \psi_1, x \models \psi_2 \in S$. By induction, we have $\mathfrak{M}_S, x \models \psi_1$ and $\mathfrak{M}_S, x \models \psi_2$ holds and hence $\mathfrak{M}_S, x \models \psi_1 \wedge \psi_2$. The case $x \models \psi_1 \vee \psi_2 \in S$ can be handled analogously.

- $x \models \langle R \rangle_{\geq n} \psi \in S$ implies $\sharp R^S(x, \psi) \geq n$ because otherwise the $\rightarrow_\geq$-rule would be applicable and S would not be complete. By induction, we have $\mathfrak{M}_S, y \models \psi$ for each y with $y \models \psi \in S$. Hence $\sharp R^{\mathfrak{M}_S}(x, \psi) \geq n$ and thus $\mathfrak{M}_S, x \models \langle R \rangle_{\geq n} \psi$.

- $x \models \langle R \rangle_{\leq n} \psi \in S$ implies $\sharp R^S(x, \psi) \leq n$ because S is clash-free. Hence it is sufficient to show that $\sharp R^{\mathfrak{M}_S}(x, \psi) \leq \sharp R^S(x, \psi)$ holds. On the contrary, assume $\sharp R^{\mathfrak{M}_S}(x, \psi) > \sharp R^S(x, \psi)$ holds. Then there is a variable y such that $Rxy \in S$ and $\mathfrak{M}_S, y \models \psi$ while $y \models \psi \notin S$. For each variable y with $Rxy \in S$ either $y \models \psi \in S$ or $y \models \sim\psi \in S$. This implies $y \models \sim\psi \in S$ and, by the induction hypothesis, $\mathfrak{M}_S, y \models \sim\psi$ holds which is a contradiction. ∎

The following theorem is an immediate consequence of Lemma 4.3, 4.6, and 4.7:

COROLLARY 4.8
The optimised algorithm is a non-deterministic decision procedure for $\text{SAT}(\mathbf{Gr}(\mathbf{K}_\mathcal{R}))$.

4.2 Complexity of the optimised algorithm

The optimised algorithm will enable us to prove Theorem 4.1. We will give a proof by sketching an implementation of this algorithm that runs in polynomial space.

LEMMA 4.9
The optimised algorithm can be implemented in PSPACE

PROOF. Let ϕ be the $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$-formula to be tested for satisfiability. We can assume ϕ to be in NNF because the transformation of a formula to NNF can be performed in linear time and space.

The key idea for the PSPACE implementation is the *trace technique* [SSS91], i.e., it is sufficient to keep only a single path (a trace) of $G(S)$ in memory at a given stage if the c.s. is generated in a depth-first manner. This has already been the key to a PSPACE upper bound for $\mathbf{K}_\mathcal{R}$ and $\mathcal{ALC}$ in [Lad77, SSS91, HM92]. To do this we need to store the values for $\sharp R^S(x, \psi)$ for each variable x in the path, each R which appears in *clos*(ϕ) and each $\psi \in$ *clos*(ϕ). By storing these values in binary form, we are able to keep information *about* exponentially many successors in memory while storing only a single path at a given stage.

Consider the algorithm in Fig. 5, where $\mathcal{R}_\phi$ denotes the set of relation names that appear in *clos*(ϕ). It re-uses the space needed to check the satisfiability of a successor y of x once the existence of a complete and clash-free "subtree" for the constraints on y has been established. This is admissible since the optimised rules will never modify this subtree once is it completed. Neither do constraints in this subtree have an influence on the completeness or the existence of a clash in the rest of the tree, with the exception that constraints of the form $y \models \psi$ for R-successors y of x contribute to the value of $\sharp R^S(x, \psi)$. These numbers play a role both in the definition of a clash and for the applicability of the $\rightarrow_\geq$-rule. Hence, in order to re-use the space occupied by the subtree for y, it is necessary and sufficient to store these numbers.

Let us examine the space usage of this algorithm. Let $n = |\phi|$. The algorithm is designed to keep only a single path of $G(S)$ in memory at a given stage. For each variable x on a path, constraints of the form $x \models \psi$ have to be stored for formulae $\psi \in$ *clos*(ϕ). The size of *clos*(ϕ) is bounded by $2n$ and hence the constraints for a single variable can be stored in $\mathcal{O}(n)$ bits. For each variable, there are at most $|\mathcal{R}_\phi| \times |$*clos*$(\phi)| = \mathcal{O}(n^2)$ counters to be stored. The numbers to be stored in these counters do not exceed the out-degree of x, which, by Lemma 4.2, is bounded by $|$*clos*$(\phi)| \times 2^{|\phi|}$. Hence each counter can be stored using $\mathcal{O}(n^2)$ bits when binary coding is used to represent the counters, and all counters for a single variable

$\mathbf{Gr}(\mathbf{K}_{\mathcal{R}}) - \mathrm{SAT}(\phi) := \mathtt{sat}(x_0, \{x_0 \models \phi\})$

```
sat(x, S):
    allocate counters ♯R^S(x, ψ) := 0 for all R ∈ R_φ and ψ ∈ clos(φ).
    while (the →∧- or the →∨-rule can be applied) and (S is clash-free) do
        apply the →∧- or the →∨-rule to S.
    od
    if S contains a clash then return "not satisfiable".
    while (the →≥-rule applies to x in S) do
        S_new := {Rxy, y ⊨ φ', y ⊨ χ_1, ..., y ⊨ χ_k}
        where
            y is a fresh variable,
            x ⊨ ⟨R⟩_{≥n} φ' triggers an application of the →≥-rule,
            {ψ_1, ..., ψ_k} = {ψ | x ⊨ ⟨R⟩_{⋈n} ψ ∈ S}, and
            χ_i is chosen non-deterministically from {ψ_i, ∼ψ_i}
        for each y ⊨ ψ ∈ S_new do increment ♯R^S(x, ψ)
        if x ⊨ ⟨R⟩_{≤m} ψ ∈ S and ♯R^S(x, ψ) > m then return "not satisfiable".
        if sat(y, S_new) = "not satisfiable" then return "not satisfiable"
    od
    remove the counters for x from memory.
    return "satisfiable"
```

Figure 5: A non-deterministic PSPACE decision procedure for $\mathrm{SAT}(\mathbf{Gr}(\mathbf{K}_{\mathcal{R}}))$.

require $\mathcal{O}(n^4)$ bits. Due to Lemma 4.2, the length of a path is limited by n, which yields an overall memory consumption of $\mathcal{O}(n^5 + n^2)$. ∎

Theorem 4.1 now is a simple Corollary from the PSPACE-hardness of $\mathbf{K}_{\mathcal{R}}$, Lemma 4.9, and Savitch's Theorem [Sav70].

5 Extensions of the Language

It is possible to extend the language $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ without loosing the PSPACE property of the satisfiability problem. In this section we extend the techniques to obtain a PSPACE algorithm for the logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$, which extends $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ by intersection of accessibility relations and inverse relations. These extension are mainly motivated from the world of Description Logics, where they are commonly studied. In this context, the logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$ can be perceived as a notational variant of the Description Logic $\mathcal{ALCQIR}$.

DEFINITION 5.1 (SYNTAX AND SEMANTICS OF $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$)
Let $\mathcal{P} = \{p_0, p_1, \dots\}$ be a set of proposition letters and let $\mathcal{R}$ be a set of *relation names*. The set $\overline{\mathcal{R}} := \mathcal{R} \cup \{R^{-1} | R \in \mathcal{R}\}$ is called the set of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-relations.

The set of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-*formulae* is the smallest set such that

1. every proposition letter is a $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-formula and,

2. if ϕ, ψ_1, ψ_2 are formulae, $n \in \mathbb{N}$, and $R_1, \ldots, R_k$ are (possibly inverse) $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-relations, then $\neg\phi$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\langle R_1 \cap \cdots \cap R_k \rangle_{\leq n}\phi$, and $\langle R_1 \cap \cdots \cap R_k \rangle_{\geq n}\phi$ are $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-formulae.

The semantics are extended accordingly:

$$\mathfrak{M}, x \models \langle R_1 \cap \cdots \cap R_k \rangle_{\leq n}\phi \text{ iff } \sharp(R_1 \cap \cdots \cap R_k)^{\mathfrak{M}}(x, \phi) \leq n$$
$$\mathfrak{M}, x \models \langle R_1 \cap \cdots \cap R_k \rangle_{\geq n}\phi \text{ iff } \sharp(R_1 \cap \cdots \cap R_k)^{\mathfrak{M}}(x, \phi) \geq n$$

where

$$\sharp(R_1 \cap \cdots \cap R_k)^{\mathfrak{M}}(x, \phi) = |\{y \in W^{\mathfrak{M}} \mid (x, y) \in R_1^{\mathfrak{M}} \cap \cdots \cap R_k^{\mathfrak{M}} \text{ and } \mathfrak{M}, y \models \phi\}|,$$

and, for $R \in \mathcal{R}$, we define

$$(R^{-1})^{\mathfrak{M}} := \{(y, x) \mid (x, y) \in R^{\mathfrak{M}}\}.$$

We will use the letters ω, σ to range over intersections of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-relations. By abuse of notation we will sometimes identify an intersection of relations ω with the set of relations occurring in it and write $R \in \omega$ iff $\omega = R_1 \cap \cdots \cap R_k$ and there is some $1 \leq i \leq k$ with $R = R_i$. To avoid dealing with relations of the form $(R^{-1})^{-1}$ we use the convention that $(R^{-1})^{-1} = R$ for any $R \in \mathcal{R}$.

Obviously every $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ formula is also a $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$ formula. Using standard bisimluation arguments one can show that $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$ is strictly more expressive than $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$.

5.1 Reasoning for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$

We will use similar techniques as in the previous section to obtain a PSPACE-algorithm for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$. The definition of a constraint system remains unchanged, but we additionally require that, for any $R \in \mathcal{R}$, a c.s. S contains the constraint 'Rxy' iff it contains the constraint '$R^{-1}yx$'. For a c.s. S, an intersection of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-relations $\omega = R_1 \cap \cdots \cap R_k$, and a formula ϕ, let $\sharp\omega^S(x, \phi)$ be the number of variables y such that $\{R_1xy, \ldots, R_kxy, y \models \phi\} \subseteq S$.

We modify the definition of *clash* to deal with intersection of relations as follows. A c.s. S contains a clash iff

- $\{x \models p, x \models \neg p\} \subseteq S$ for some variable x and a proposition letter p, or
- $x \models \langle \omega \rangle_{\leq n}\phi \in S$ and $\sharp\omega^S(x, \phi) > n$ for some variable x, intersection of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-relations ω, formula ϕ and $n \in \mathbb{N}$.

The set of rules dealing with the extended logic is shown in Figure 6. We require the algorithm to maintain a binary relation $\prec_S$ between the variables in a c.s. S with $x \prec_S y$ iff y was inserted by the $\rightarrow_{\geq}$-rule to satisfy a constraint for x. When considering the graph $G(S)$, the relation $\prec_S$ corresponds to the successor relation between nodes. Hence, when $x \prec_S y$ holds we will call y a successor of x and x a predecessor of y. We denote the transitive closure of $\prec_S$ by $\prec_S^+$. For a set of variables $\mathcal{X}$ and a c.s. S, we denote the subset of S in which no variable from $\mathcal{X}$ occurs in a constraint by $S - \mathcal{X}$. The $\rightarrow_{\wedge}$-, $\rightarrow_{\vee}$- and $\rightarrow_{\text{choose}}$-rule are called "non-generating rules" while the $\rightarrow_{\geq}$-rule is called a "generating rule". The algorithm which uses these rules will be called the $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-algorithm.

$\rightarrow_\wedge$-, $\rightarrow_\vee$-rule: see Fig. 1

$\rightarrow_{\text{choose}}$-rule: if 1. $x \models \langle\omega\rangle_{\bowtie n}\phi \in S$ and
2. for some $R \in \omega$ there is a y with $Rxy \in S$, and
$\{y \models \phi, y \models \sim\phi\} \cap S = \emptyset$
then $S \rightarrow_{\text{choose}} S' \cup \{y \models \chi\}$ where $\chi \in \{\phi, \sim\phi\}$
and $S' = S - \{z \mid y \prec_S^+ z\}$

$\rightarrow_\geq$-rule: if 1. $x \models \langle\omega\rangle_{\geq n}\phi \in S$, and
2. $\sharp\omega^S(x, \phi) < n$, and
3. no non-generating rule can be applied to a constraint for x
then $S \rightarrow_\geq S \cup \{y \models \psi\} \cup S' \cup S''$ and set $x \prec_S y$ where
$S' = \{y \models \chi_1, \ldots, y \models \chi_k\}$, $\chi_i \in \{\psi_i, \sim\psi_i\}$, and
$\{\psi_1, \ldots, \psi_k\} = \{\psi \mid x \models \langle\sigma\rangle_{\bowtie m}\psi \in S\}$
$S'' = \{R_1xy, R_1^{-1}yx, \ldots, R_mxy, R_m^{-1}yx\}$ and
$\omega \subseteq \{R_1, \ldots, R_m\} \subseteq \overline{\mathcal{R}}$
y is a fresh variable

Figure 6: The completion rules for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$.

The $\rightarrow_\geq$-rule, while looking complicated, is a straightforward extension of the $\rightarrow_\geq$-rule for $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$, which takes into account that we also need to guess additional *relations* between the old variable x and the freshly introduced variable y. The $\rightarrow_{\text{choose}}$-rule requires more explanation.

For $\mathbf{Gr}(\mathbf{K}_\mathcal{R})$, the optimised algorithm generates a c.s. S in a way that, whenever $x \models \langle R\rangle_{\bowtie n}\psi \in S$, then, for any y with $Rxy \in S$, either $y \models \psi \in S$ of $y \models \sim\psi \in S$. This was achieved by suspending the generation of any successors y of x until S contained all constraints of the from $x \models \phi$ it would ever contain. In the presence of inverse relations, this is no longer possible because y might be generated as a predecessor of x and hence before it was possible to know which ψ might be relevant. There are at least two possible ways to overcome this problem. One is, to guess, for every x and *every* $\psi \in$ *clos*(ϕ), whether $x \models \psi$ or $x \models \sim\psi$. In this case, since the termination of the optimised algorithm as shown in Lemma 4.3 relies on the fact that the modal depth strictly decreases along a path in the induced graph $G(S)$, termination would no longer be guaranteed. It would have to be enforced by different means.

Here, we use another approach. We can distinguish two different situations where $\{x \models \langle\omega\rangle_{\bowtie n}\psi, Rxy\} \subseteq S$ for some $R \in \omega$, and $\{y \models \psi, y \models \sim\psi\} \cap S = \emptyset$, namely, whether y is a predecessor of x ($y \prec_S x$) or a successor of x ($x \prec_S y$). The second situation will never occur. This is due to the interplay of the $\rightarrow_\geq$-rule, which is suspended until all known relevant information has been added for x, and the $\rightarrow_{\text{choose}}$-rule, which deletes certain parts of the c.s. whenever new constraints have to be added for predecessor variables.

The first situation is resolved by non-deterministically adding either $y \models \psi$ or $y \models \sim\psi$ to S. The subsequent deletion of all constraints involving variables from $\{z \mid y \prec_S^+ z\}$, which corresponds to all subtrees of $G(S)$ rooted at successors of y, is necessary to make this rule "compatible" with the trace-technique we want to employ in order to obtain a PSPACE-algorithm. The correctness of the trace-approach relies on the property that, once we have established the existence of a complete and clash-free "subtree" for a node x, we can remove

$$
\begin{aligned}
\{x \models \phi\} &\rightarrow_{\wedge} \dots \\
&\rightarrow_{\wedge} \underbrace{\{x \models \phi, x \models \langle R_1\rangle_{\leq 0} q, x \models \langle R_1\rangle_{\geq 1}(p \vee q), x \models \langle R_2\rangle_{\geq 1}\langle R_2^{-1}\rangle_{\leq 0}\langle R_1\rangle_{\geq 1} p\}}_{S_1} \\
&\rightarrow_{\geq} \underbrace{S_1 \cup \{R_1 xy, R_1^{-1} yx, y \models (p \vee q), y \models \neg q\}}_{S_2} \rightarrow_{\vee} \underbrace{S_2 \cup \{y \models p\}}_{S_3}
\end{aligned}
$$

Figure 7: Inverse roles make tracing difficult.

this tree from memory because it will not be modified by the algorithm. In the presence of inverse relations this can be no longer taken for granted as can be shown by the formula

$$\phi = \langle R_1\rangle_{\leq 0} q \ \wedge\ \langle R_1\rangle_{\geq 1}(p \vee q) \ \wedge\ \langle R_2\rangle_{\geq 1}\langle R_2^{-1}\rangle_{\leq 0}\langle R_1\rangle_{\geq 1} p$$

Figure 7 shows the beginning of a run of the algorithm for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$. After a number of steps, a successor y of x has been generated and the expansion of constraints has produced a complete and clash-free subtree for y. Nevertheless, the formula ϕ is not satisfiable. The expansion of $\langle R_2\rangle_{\geq 1}\langle R_2^{-1}\rangle_{\leq 0}\langle R_1\rangle_{\geq 1} p$ will eventually lead to the generation of the constraint $x \models \sim\langle R_1\rangle_{\geq 1} p = \langle R_1\rangle_{\leq 0} p$, which clashes with $y \models p$. If the subtree for y would already have been deleted from memory, this clash would go undetected. For this reason, the $\rightarrow_{\text{choose}}$-rule deletes all successors of the modified node, which, while duplicating some work, makes it possible to detect these clashes even when tracing through the c.s. A similar technique has been used in [HST99] to obtain a PSPACE-result for a Description Logic with inverse roles.

5.2 Correctness of the Algorithm

As for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$, we have to show termination, soundness, and correctness of the algorithm for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$.

5.2.1 Termination

Obviously, the deletion of constraints in S makes a new proof of termination necessary, since the proof of Lemma 4.3 relied on this fact. Please note, that the Lemma 4.2 still holds for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$.

LEMMA 5.2 (TERMINATION)
Any sequence of rule applications starting from a c.s. $S = \{x_0 \models \phi\}$ of the $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$ algorithm is finite.

PROOF. The sequence of rule applications induces a sequence of trees. As before, the depth and out-degree of this tree is bounded in $|\phi|$ by Lemma 4.2. For each variable x, $\mathcal{L}(x)$ is a subset of the finite set *clos*(ϕ). Each application of a rule either

- adds a constraint of the form $x \models \psi$ and hence adds an element to $\mathcal{L}(x)$, or
- adds fresh variables to S and hence adds additional nodes to the tree $G(S)$, or

- adds a constraint to a node y and deletes all subtrees rooted at successors of y.

Assume that algorithm does not terminate. Due to the mentioned facts this can only be because of an infinite number of deletions of subtrees. Each node can of course only be deleted once, but the successors of a single node may be deleted several times. The root of the completion tree cannot be deleted because it has no predecessor. Hence there are nodes which are never deleted. Choose one of these nodes y with maximum distance from the root, i.e., which has a maximum number of ancestors in $\prec_S$. Suppose that y's successors are deleted only finitely many times. This can not be the case because, after the last deletion of y's successors, the "new" successors were never deleted and thus y would not have maximum distance from the root. Hence y triggers the deletion of its successors infinitely many times. However, the $\rightarrow_{\text{choose}}$-rule is the only rule that leads to a deletion, and it simultaneously leads to an increase of $\mathcal{L}(y)$, namely by the missing concept which caused the deletion of y's successors. This implies the existence of an infinitely increasing chain of subsets of *clos*(ϕ), which is clearly impossible. ■

5.2.2 Soundness and Completeness

LEMMA 5.3 (SOUNDNESS)
Let ϕ be a $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-formula in NNF. If the completion rules can be applied to $\{x_0 \models \phi\}$ such that they yield a complete and clash-free c.s., then $\phi \in \text{SAT}(\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}}))$.

PROOF. Let S be a complete and clash-free c.s. obtained by a sequence of rule applications from $\{x_0 \models \phi\}$. We show that the canonical structure $\mathfrak{M}_S$ is indeed a model of ϕ, where the canonical structure for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$ is defined as in Definition 3.2. Please note, that we need the condition "$Rxy \in S$ iff $R^{-1}yx \in S$" to make sure that all information from the c.s. is reflected in the canonical structure.

By induction over the norm of formulae $\|\psi\|$ as defined in the proof of Lemma 4.7, we show that, for a complete and clash-free c.s. S, $x \models \psi \in S$ implies $\mathfrak{M}_S, x \models \psi$. The only interesting cases are when ψ starts with a modal operator.

- $x \models \langle\omega\rangle_{\geq n}\psi \in S$ implies $\omega^S(x,\psi) \geq n$ because S is complete. Hence, there are n distinct variables $y_1, \ldots, y_n$ with $y_i \models \psi \in S$ and $Rxy_i \in S$ for each $1 \leq i \leq n$ and $R \in \omega$. By induction, we have $\mathfrak{M}_S, y_i \models \psi$ and $(x, y_i) \in \omega^{\mathfrak{M}_S}$ and hence $\mathfrak{M}_S, x \models \langle\omega\rangle_{\geq n}\psi$.

- $x \models \langle\omega\rangle_{\leq n}\psi \in S$ implies, for any $R \in \omega$ and any y with $Rxy \in S$, $y \models \psi \in S$ or $y \models {\sim}\psi \in S$. For any predecessor of x, this is guaranteed by the $\rightarrow_{\text{choose}}$-rule, for any successor of x by the $\rightarrow_{\geq}$-rule which is suspended until no non-generating rule rules can applied to x or any predecessor of x together with the reset-restart mechanism that is triggered by constraints "moving upwards" from a variable to its predecessor.

  We show that $\sharp\omega^{\mathfrak{M}_S}(x,\psi) \leqslant \sharp\omega^S(x,\psi)$: assume $\sharp\omega^{\mathfrak{M}_S}(x,\psi) > \sharp\omega^S(x,\psi)$. This implies the existence of some y with $(x,y) \in R^{\mathfrak{M}_S}$ for each $R \in \omega$ and $\mathfrak{M}_S, y \models \psi$ but $y \models \psi \notin S$. This implies $y \models {\sim}\psi \in S$, which, by induction yields $\mathfrak{M}_S, y \models {\sim}\psi$ in contradiction to $\mathfrak{M}_S, y \models \psi$.

Since constraints for the initial variable x_0 are never deleted from S, we have that $x_0 \models \phi \in S$ and hence $\mathfrak{M}_S, x_0 \models \phi$ and $\phi \in \text{SAT}(\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}}))$. ■

The following lemma combines the local and global completeness proof for the $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-algorithm

LEMMA 5.4 (COMPLETENESS)
If $\phi \in$ SAT($\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$) in NNF, then there is a sequence of the $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-rule starting with $S = \{x_0 \models \phi\}$ that results in a complete and clash-free c.s.

PROOF. Let $\mathfrak{M}$ be a model for ψ and $\overline{\mathcal{R}}_\phi$ the set of relations that occur in ϕ together with their inverse. We use $\mathfrak{M}$ to guide the application of the non-deterministic completion rules by incremently defining a function α mapping variables from the c.s. to elements of $W^{\mathfrak{M}}$. The function α will always satisfy the following conditions:

$$\left.\begin{array}{l} \text{1. if } x \models \psi \in S \text{ then } \mathfrak{M}, \alpha(x) \models \psi \\ \text{2. if } Rxy \in S \text{ then } \{R \mid Rxy \in S\} = \{R \mid (\alpha(x), \alpha(y)) \in R^{\mathfrak{M}}\} \cap \overline{\mathcal{R}}_\phi \\ \text{3. if } y, z \text{ are distinct variables such that } \{R_1xy, R_2xz\} \subseteq S, \text{ then } \alpha(y) \neq \alpha(z) \end{array}\right\} (*)$$

CLAIM: Whenever $(*)$ holds for a c.s. S and a function α and a rule is applicable to S then it can be applied in a way that maintains $(*)$.

- The $\to_\wedge$-rule: if $x \models \psi_1 \wedge \psi_2 \in S$, then $\mathfrak{M}, \alpha(x) \models (\psi_1 \wedge \psi_2)$. This implies $\mathfrak{M}, \alpha(x) \models \psi_i$ for $i = 1, 2$, and hence the rule can be applied without violating $(*)$.
- The $\to_\vee$-rule: if $x \models \psi_1 \vee \psi_2 \in S$, then $\mathfrak{M}, \alpha(x) \models (\psi_1 \vee \psi_2)$. This implies $\mathfrak{M}, \alpha(x) \models \psi_1$ or $\mathfrak{M}, \alpha(x) \models \psi_2$. Hence the $\to_\vee$-rule can add a constraint $x \models \chi$ with $\chi \in \{\psi_1, \psi_2\}$ such that $(*)$ still holds.
- The $\to_{\text{choose}}$-rule: obviously, either $\mathfrak{M}, \alpha(y) \models \psi$ or $\mathfrak{M}, \alpha(y) \models \sim\psi$ for any variable y in S. Hence, the rule can always be applied in a way that maintains $(*)$. Deletion of nodes does not violate $(*)$.
- The $\to_\geq$-rule: if $x \models \langle\omega\rangle_{\geq n}\phi' \in S$, then $\mathfrak{M}, \alpha(x) \models \langle\omega\rangle_{\geq n}\phi'$. This implies $\sharp\omega^{\mathfrak{M}}(\alpha(x), \phi') \geqslant n$. We claim that there is an element $t \in W^{\mathfrak{M}}$ such that

  $$\left.\begin{array}{l} (\alpha(x), t) \in R^{\mathfrak{M}} \text{ for each } R \in \omega, \text{ and } \mathfrak{M}, t \models \psi, \text{ and} \\ t \notin \{\alpha(y) \mid Rxy \in S\} \end{array}\right\} (**)$$

  We will come back to this claim later. Let $\psi_1, \ldots, \psi_k$ be an enumeration of the set $\{\psi \mid x \models \langle\sigma\rangle_{\bowtie m} \in S\}$ The $\to_\geq$-rule can add the constraints

  $$S' = \{y \models \psi_i \mid \mathfrak{M}, t \models \psi_i\} \cup \{y \models \sim\psi_i \mid \mathfrak{M}, t \not\models \psi_i\}$$
  $$S'' = \{Rxy \mid R \in \overline{\mathcal{R}}_\phi, (\alpha(x), t) \in R^{\mathfrak{M}}\} \cup \{Ryx \mid R \in \overline{\mathcal{R}}_\phi, (t, \alpha(x)) \in R^{\mathfrak{M}}\}$$

  as well as $\{y \models \phi'\}$ to S. If we set $\alpha' := \alpha[y \mapsto t]$, then the obtained c.s. together with α' satisfies $(*)$.

  Why does there exists an element t that satisfies $(**)$? Let $s \in W^{\mathfrak{M}}$ be an arbitrary element with $(\alpha(x), s) \in \omega^{\mathfrak{M}}$ and $\mathfrak{M}, s \models \psi$ that appears as an image of an arbitrary element y with $Rxy \in S$ for some $R \in \overline{\mathcal{R}}_\phi$. Condition 2 of $(*)$ implies that $Rxy \in S$ for any $R \in \omega$ and also $y \models \psi \in S$ must hold as follows:

  Assume $y \models \psi \notin S$. This implies $y \models \sim\psi \in S$: either $y \prec_S x$, then in order for the $\to_\geq$-rule to be applicable, no non-generating rules and especially the $\to_{\text{choose}}$-rule is not applicable to x and its ancestor, which implies $\{y \models \psi, y \models \sim\psi\} \cap S \neq \emptyset$. If not $y \prec_S x$ then y must have been generated by an application of the $\rightarrow_{\geq}$-rule to x. In order for this rule to be applicable no non-generating rule may have been applicable to x or any of its ancestors. This implies that at the time of the generation of y already $x \models \langle\omega\rangle_{\geq n}\psi \in S$ held and hence the $\rightarrow_{\geq}$-rule ensures $\{y \models \psi, y \models \sim\psi\} \cap S \neq \emptyset$.

In any case $y \models \sim\psi \in S$ holds and together with Condition 1 of $(*)$ this implies $\mathfrak{M}, s \not\models \psi$ which contradicts $\mathfrak{M}, s \models \psi$.

Together this implies that, whenever an element s with $(\alpha(x), s) \in \omega^{\mathfrak{M}}$ and $\mathfrak{M}, s \models \psi$ is assigned to a variable y with $Rxy \in S$, then it must be assigned to a variable that contributes to $\sharp\omega^S(x, \psi)$. Since the $\rightarrow_{\geq}$-rule is applicable there are less than n such variables and hence there must be an unassigned element t as required by $(**)$.

This concludes the proof of the claim. The claim yields the lemma as follows: obviously, $(*)$ holds for the initial c.s. $\{x_0 \models \phi\}$, if we set $\alpha(x_0) := s_0$ for an element s_0 with $\mathfrak{M}, s_0 \models \phi$ (such an element must exist because $\mathfrak{M}$ is a model for ϕ). The claim implies that, whenever a rule is applicable, then it can be applied in a manner that maintains $(*)$. Lemma 5.2 yields that each sequence of rule applications must terminate, and also each c.s. for which $(*)$ holds is necessarily clash-free. It cannot contain a clash of the form $\{x \models p, x \models \neg p\} \subseteq S$ because this would imply $\mathfrak{M}, \alpha(x) \models p$ and $\mathfrak{M}, \alpha(x) \not\models p$. It can neither contain a clash of the form $x \models \langle\omega\rangle_{\leq n}\psi \in S$ and $\sharp\omega^S(x, \psi) > n$ because α is an injective function on $\{y \mid Rxy \in S\}$ and preserves all relations in $\overline{\mathcal{R}}_\phi$. Hence $\sharp\omega^S(x, \psi) > n$ implies $\sharp\omega^{\mathfrak{M}}(x, \psi) > n$, which cannot be the case since $\mathfrak{M}, \alpha(x) \models \langle\omega\rangle_{\leq n}\psi$. ∎

As a corollary of Lemma 5.2, 5.3, and 5.4 we get:

COROLLARY 5.5
The $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-algorithm is a non-deterministic decision procedure for SAT$(\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}}))$.

5.3 Complexity of the Algorithm

As for the optimised algorithm for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$, we have to show that the $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-algorithm can be implemented in a way that consumes only polynomial space. This is done similarly to the $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$-case, but we have to deal with two additional problems: we have to find a way to implement the "reset-restart" caused by the $\rightarrow_{\text{choose}}$-rule, and we have to store the values of the relevant counters $\omega^S(x, \psi)$. It is impossible to store the values for each possible intersection of relations ω because the are exponentially many of these. Fortunately, storing only the values for those ω which actually appear in ϕ is sufficient.

LEMMA 5.6
The $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-algorithm can be implemented in PSPACE.

PROOF. Consider the algorithm in Figure 8, where Ω_ϕ denotes all intersections of relations that occur in ϕ. As the algorithm for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$, it re-uses the space used to check for the existence of a complete and clash-free "subtree" for each successor y of a variable x. Counter variables are used to keep track of the values $\sharp\omega^S(x, \psi)$ for all relevant ω and ψ. This can be done in polynomial space. Resetting a node and restarting the generation of its successors is achieved by resetting all successor counters. Please note, how the predecessor of a node is taken into account when initialising the counter variables.

Since the length of paths in a c.s. is polynomial bounded in $|\phi|$ and all necessary book-keeping information can be stored in polynomial space, this proves the lemma. ∎

Obviously, SAT($\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$) is PSPACE-hard, hence Lemma 5.6 and Savitch's Theorem [Sav70] yield:

THEOREM 5.7
Satisfiability for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$ is PSPACE-complete if the numbers in the input are represented using binary coding.

As a simple corollary, we get the solution of an open problem in [DLNN97]:

COROLLARY 5.8
Satisfiability for $\mathcal{ALCNR}$ is PSPACE-complete if the numbers in the input are represented using binary coding.

PROOF. The DL $\mathcal{ALCNR}$ is a syntactic restriction of the DL $\mathcal{ALCQIR}$, which, in turn, is a syntactical variant of $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$. Hence, the $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$-algorithm can immediately be applied to $\mathcal{ALCNR}$-concepts. ∎

6 Conclusion

We have shown that by employing a space efficient tableaux algorithm satisfiability of the logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ can be decided in PSPACE, which is an optimal result with respect to worst-case complexity. Moreover, we have extended the technique to the logic $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$, which extends $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ both by inverse relations and intersection of relations. This logic is a notational variant of the Description Logic $\mathcal{ALCQIR}$, for which the complexity of concept satisfiability has also been open. This settles the complexity of the DL $\mathcal{ALCNR}$ for which the upper complexity bound with binary coding had also been an open problem [DLNN97]. While the algorithms presented in this work certainly are only optimal from the viewpoint of worst-case complexity, they are relatively simple and will serve as the starting-point for a number of optimisations leading to more practical implementations. They also serve as tools to establish the upper complexity bound of the problems and thus shows that tableaux based reasoning for $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}})$ and $\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}})$ can be done with optimum worst-case complexity. This establishes a kind of "theoretical benchmark" that all algorithmic approaches can be measured against.

Acknowledgments.

I would like to thank Franz Baader, Ulrike Sattler, and an anonymous referee for valuable comments and suggestions. Part of this work was supported by the DFG, Project No. GR 1324/3-1.

$\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}}) - \mathrm{SAT}(\phi) := \mathtt{sat}(x_0, \{x_0 \models \phi\})$

$\mathtt{sat}(x, S)$:

  allocate counters $\sharp\omega^S(x,\psi)$ for all $\omega \in \Omega_\phi$ and $\psi \in$ *clos*(ϕ).

 restart:

  `for each` counter $\sharp\omega^S(x,\psi)$:

    `if` x has a predecessor $y \prec_S x$ `and` $\omega \subseteq \{R \mid Rxy \in S\}$ `and` $y \models \psi \in S$

      `then` $\sharp\omega^S(x,\psi) := 1$ `else` $\sharp\omega^S(x,\psi) := 0$

  `while` (the $\rightarrow_\wedge$- or the $\rightarrow_\vee$-rule can be applied at x) `and` (S is clash-free) `do`

    apply the $\rightarrow_\wedge$- or the $\rightarrow_\vee$-rule to S.

  `od`

  `if` S contains a clash `then return` "not satisfiable".

  `if` the $\rightarrow_{\text{choose}}$-rule is applicable to the constraint $x \models \langle\omega\rangle_{\bowtie n}\psi \in S$

    `then return` "restart with ψ"

  `while` (the $\rightarrow_\geq$-rule applies to a constraint $x \models \langle\omega\rangle_{\geq n}\phi' \in S$) `do`

    $S_{\text{new}} := \{y \models \phi'\} \cup S' \cup S''$

    `where`

      y is a fresh variable

      $\{\psi_1, \ldots, \psi_k\} = \{\psi \mid x \models \langle\sigma\rangle_{\bowtie m}\psi \in S\}$

      $S' = \{y \models \chi_1, \ldots, y \models \chi_k\}$, and

      χ_i is chosen non-deterministically from $\{\psi_i, {\sim}\psi_i\}$

      $S'' = \{R_1xy, R_1^{-1}yx, \ldots, R_lxy, R_l^{-1}yx\}$

      $\{R_1, \ldots, R_l\}$ is chosen non-deterministically with $\omega \subseteq \{R_1, \ldots, R_l\} \subseteq \overline{\mathcal{R}}_\phi$

    `for each` ψ with $y \models \psi \in S'$ `and` $\sigma \in \Omega_\phi$ with $\sigma \subseteq \{R \mid Rxy \in S''\}$ `do`

      increment $\sharp\sigma^S(x,\psi)$

    `if` $x \models \langle\sigma\rangle_{\leq m}\psi \in S$ and $\sharp\sigma^S(x,\psi) > m$

      `then return` "not satisfiable".

    $result := \mathtt{sat}(y, S \cup S_{\text{new}})$

    `if` $result =$ "not satisfiable" `then return` "not satisfiable"

    `if` $result =$ "restart with ψ" `then`

      $S := S \cup \{x \models \chi\}$

      `where` χ is chose non-deterministically from $\{\psi, {\sim}\psi\}$

      `goto` restart

  `od`

  remove the counters for x from memory.

  `return` "satisfiable"

Figure 8: A non-deterministic PSPACE decision procedure for $\mathrm{SAT}(\mathbf{Gr}(\mathbf{K}_{\mathcal{R}_\cap^{-1}}))$.

References

[AvBN98] H. Andréka, J. van Benthem, and I. Németi Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[BBH96] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[CLN94] D. Calvanese, M. Lenzerini, and D. Nardi. A Unified Framework for Class Based Representation Formalisms. *Proc. of KR-94*, 1994.

[dHR95] W. Van der Hoek and M. De Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, June 1995.

[DLNN97] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 10 April 1997.

[Fin72] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logic*, 13:516–520, 1972.

[GS96] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures—the case study of modal K. *Proc. of CADE-13*, LNCS 1104. Springer, 1996.

[HB91] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR-91*, pages 335–346, Boston (USA), 1991.

[HM92] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for model logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, April 1992.

[HS97] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In *Proc. of IJCAI-97)*, volume 1, pages 202–207, 1997.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In H. Ganzinger and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)* LNAI number 1705, Springer-Verlag.

[Lad77] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, September 1977.

[OS97] H. J. Ohlbach and R. A. Schmidt. Functional translation and second-order frame properties of modal logics. *Journal of Logic and Computation*, 7(5):581–603, October 1997.

[OSH96] H. J. Ohlbach, R. A. Schmidt, and U. Hustadt. Translating graded modalities into predicate logic. In H. Wansing, editor, *Proof Theory of Modal Logic*, volume 2 of *Applied Logic Series*, pages 253–291. Kluwer, 1996.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, April 1970.

[Sch91] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, 1991.

[Sch97] R. A. Schmidt. Resolution is a decision procedure for many propositional modal logics: Extended abstract. In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic '96*. CLSI Publications, 1997.

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[Tob99] S. Tobies. A PSpace algorithm for graded modal logic. In H. Ganzinger, editor, *Automated Deduction – CADE-16, 16th International Conference on Automated Deduction*, LNAI 1632, pages 52–66, Trento, Italy, July 7–10, 1999. Springer-Verlag.