ExpTime Tableau Decision Procedures for Regular Grammar Logics with Converse [⋆]

Linh Anh Nguyen[1] and Andrzej Szałas[1,2]

[1] Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{nguyen,andsz}@mimuw.edu.pl

[2] Department of Computer and Information Science, Linköping University
SE-581 83 Linköping, Sweden

Abstract. Grammar logics were introduced by Fariñas del Cerro and Penttonen in [7] and have been widely studied. In this paper we consider regular grammar logics with converse (*REG^c* logics) and present sound and complete tableau calculi for the general satisfiability problem of *REG^c* logics and the problem of checking consistency of an ABox w.r.t. a TBox in a *REG^c* logic. Using our calculi we develop ExpTime (optimal) tableau decision procedures for the mentioned problems, to which various optimization techniques can be applied. We also prove a new result that the data complexity of the instance checking problem in *REG^c* logics is coNP-complete.

1 Introduction

A *grammar logic* is a normal multimodal logic characterized by inclusion axioms of the form $[\sigma_1]\dots[\sigma_h]\varphi \to [\varrho_1]\dots[\varrho_k]\varphi$, where $[\sigma_i]$ and $[\varrho_j]$ are universal modal operators indexed by indices σ_i and ϱ_j from some set Σ. Inclusion axioms give rise to production rules of the form $\sigma_1 \dots \sigma_h \to \varrho_1 \dots \varrho_k$. When the rules are restricted to have only one symbol at the left hand side, the considered logic is called a *context-free grammar logic*. If, for each $\sigma \in \Sigma$, the set of words derivable from σ using the production rules is a regular language explicitly specified by a finite automaton then the considered logic is called a *regular grammar logic*.

Assume now that there is a symmetric one-to-one map on Σ that associates each $\sigma \in \Sigma$ with $\overline{\sigma} \in \Sigma$, intuitively standing for the *converse* of σ. If the set S of production rules of the considered regular grammar logic is symmetric in the sense that $(\sigma \to \varrho_1 \dots \varrho_k) \in S$ iff $(\overline{\sigma} \to \overline{\varrho}_k \dots \overline{\varrho}_1) \in S$, then the logic is a *regular grammar logic with converse*.

Grammar logics have been introduced by Fariñas del Cerro and Penttonen in [7] and have been studied widely, e.g., in [1,3,5,11,23]. In [1], Baldoni et al. gave a prefixed tableau calculus for grammar logics and used it to show that the general (uniform) satisfiability problem[3] of right linear grammar logics is decidable and that the general satisfiability problem of context-free grammar logics is undecidable. In [3], by using a transformation into the satisfiability problem of propositional dynamic logic (PDL), Demri proved that the general satisfiability problem of regular grammar logics is ExpTime-complete. In [5], Demri and de Nivelle gave a translation of the satisfiability problem of grammar logics with converse into the two-variable guarded fragment of first-order logic and have

⋆ This work is an extension of the conference paper [25]. It is supported by grants N N206 3982 33 and N N206 399334 from the Polish Ministry of Science and Higher Education.

[3] i.e., the satisfiability problem, where a specification of the logic is also given as an input of the problem

shown that the general satisfiability problem of regular grammar logics with converse is also ExpTime-complete. In [11], Goré and Nguyen gave an ExpTime tableau decision procedure for the general satisfiability problem of regular grammar logics.

In this work we consider theorem proving in regular grammar logics with converse (REG^c logics). The class of these logics is large and contains many common and useful modal logics. Here are some examples (see also [5]):

– All 15 basic monomodal logics obtained from K by adding an arbitrary combination of axioms D, T, B, 4, 5 are REG^c logics.[4] The multimodal versions of these logics are also REG^c logics.
– The description logic $\mathcal{SHI}$ and its extension with complex role inclusion axioms studied by Horrocks and Sattler in [18] are REG^c logics. The whole class of regular grammar logics has also been studied, e.g., by Nguyen [23], as a class of description logics. Note that the notion of complex role inclusion axiom given in [18] is strictly less general than the notion of inclusion axiom (of the form $[\sigma]\varphi \to [\varrho_1]\ldots[\varrho_n]\varphi$) of REG^c logics. REG^c can be treated as an extension of the description logic $\mathcal{SHI}$, which together with numeric restrictions and other concept constructors would yield very expressive description logics.
– Regular modal logics of agent beliefs studied by Goré and Nguyen in [13] are REG^c logics. Those logics use only a simple form of axiom 5 for expressing negative introspection of single agents. Axioms of REG^c can be used to express negative introspection of groups of agents.[5] However, in contrast with the logics studied in [13], cuts seem not eliminable for traditional (unlabeled) tableau calculi for the whole class REG^c.

There are two main approaches for theorem proving in modal logics: the direct approach, where one develops a theorem prover directly for the logic under consideration, and the translation-based approach, where one translates the logic into some other logic with developed proof techniques. The translation method proposed by Demri and de Nivelle [5] for REG^c logics is interesting from the theoretical point of view. It allows one to establish the complexity and sheds new light on translation approach for modal logics. On the other hand, as stated in [5], the direct approach has the advantage that a specialized algorithm can make use of specific properties of the logic under consideration, enabling optimizations that would not work in general. From the experience on optimizing tableau theorem prover TGC [24], sometimes even a minor modification may significantly increase or decrease the performance of a prover. The direct approach is therefore worth studying.

The direct approach based on tableaux has been widely applied for modal logics [26,8,27,1,10,2], because it allows to employ many useful optimization techniques (see, e.g., [17,6,24]), some of which are specific for tableaux.[6]

To our best knowledge, no tableau calculi have been developed for REG^c logics. In [4], Demri and de Nivelle gave a translation of REG^c logics into CPDL (converse PDL). One

[4] See [5] for 10 of them. For the 5 remaining logics, use $\langle\sigma\rangle\top$ as global assumptions.
[5] The general form $\neg[\sigma_1]\varphi \to [\sigma_2]\neg[\sigma_3]\varphi$ of axiom 5 can be expressed by $[\overline{\sigma}_3]\psi \to [\overline{\sigma}_1][\sigma_2]\psi$. Here, σ_1, σ_2 and σ_3 may represent groups of agents.
[6] Not all optimization techniques proposed in [17,6,24] are particularly useful. Also, they cannot be combined all together. But each of [17,6,24] proposes a number of specific good ideas for optimizing tableau decision procedures.

can use that translation together with the tableau decision procedure for CPDL given by De Giacomo and Massacci [2] for deciding REG^c logics. This method uses the translation approach and, additionally, has the disadvantage that the formal decision procedure given in [2] for CPDL has non-optimal NExpTime complexity. Although De Giacomo and Massacci [2] described also a transformation of their NExpTime algorithm into an ExpTime decision procedure for CPDL, the description is informal and unclear. Namely, the transformation is based on Pratt's global caching method formulated for PDL [26], but no global caching method has been formalized and proved sound for labeled tableaux that allow modifying labels of ancestor nodes in order to deal with converse.[7]

In this work we develop a sound and complete tableau calculus for deciding the general satisfiability problem of REG^c logics. Our calculus is an extension of the tableau calculus for regular grammar logics given by Goré and Nguyen in [11]. To deal with converse, we use an analytic cut rule. Similarly to [21,12], our cut rule is a kind of "guessing the future" for nodes in traditional (unlabeled) tableaux. Besides, there is a substantial difference comparing to [11]. Namely, Goré and Nguyen introduced only universal automaton-modal operators for regular grammar logics, while using cuts to deal with converse we have to use also existential automaton-modal operators. As a consequence, our calculus for REG^c deals also with "eventualities" (like operators $\langle\alpha^*\rangle$ of PDL). For that we adopt the tableau method given by Pratt for PDL [26], but with a more direct formulation. Our tableaux in REG^c logics are "and-or" graphs constructed using traditional tableau rules and global caching. The idea of global caching appeared in Pratt's work [26] on PDL and has been formalized and proved sound by Goré and Nguyen for traditional tableaux in a number of other modal and description logics [11,12,13,14]. Similarly as for PDL [26] but in contrast with [11,12,13,14], checking satisfiability in REG^c logics deals not only with the local consistency but also with a global consistency property of the constructed "and-or" graph.

Using our tableau calculus, we give an ExpTime (optimal) tableau decision procedure for the general satisfiability problem of REG^c logics. We also briefly discuss optimizations for the procedure.

REG^c logics can also be used as description logics. Two basic components of description logic theories are ABoxes and TBoxes. An ABox (*assertion box*) consists of facts, and a TBox (*terminological box*) consists of formulas expressing relationships between concepts. In [9], by encoding the ABox by "nominals" and "internalizing" the TBox, De Giacomo showed that the complexity of checking consistency of an ABox w.r.t. a TBox in CPDL is ExpTime-complete. Using the translation of REG^c logics into CPDL given by Demri and de Nivelle [4], one can show that the problem of checking consistency of an ABox w.r.t. a TBox in a REG^c logic is ExpTime-complete.

Extending our method to deal with ABox assertions, we give the first ExpTime tableau decision procedure not based on transformation for checking consistency of an ABox w.r.t. a TBox in a REG^c logic.

[7] According to Donini and Massacci [6, page 89], the caching optimization technique *"prunes heavily the search space but its unrestricted usage may lead to unsoundness [37]. It is conjectured that 'caching' leads to EXPTIME-bounds but this has not been formally proved so far, nor the correctness of caching has been shown."* Goré and Nguyen have recently formalized sound global caching [11,12,13,14] for *traditional* (unlabeled) tableaux, which never look back at ancestor nodes.

We also study data complexity of the instance checking problem in REG^c logics. For the well-known description logic $\mathcal{SHIQ}$, Hustadt et al. [19] proved that data complexity of that problem is coNP-complete. The lower bound for data complexity of that problem in REG^c is coNP-hard (shown for $\mathcal{ALC}$ by Schaerf in [28]). In this paper, by establishing the upper bound, we prove a new result that the data complexity of the instance checking problem in REG^c logics is coNP-complete.

The rest of this paper is structured as follows. In Section 2 we give definitions for REG^c logics. Next, in Section 3, we present our tableau calculus for the general satisfiability problem of REG^c logics. Section 4 contains proofs of its soundness and completeness. In Section 5 we present our decision procedure for the general satisfiability problem of REG^c logics. Section 6 is devoted to a study of REG^c in the context of description logics. In particular, we present there a tableau calculus and decision procedures for checking consistency of an ABox w.r.t. a TBox in a REG^c logic, and prove complexity results. Finally, Section 7 concludes this work.

2 Preliminaries

2.1 Regular Semi-Thue Systems

Let Σ^+ be a finite set of symbols. For $\sigma \in \Sigma^+$, we use $\overline{\sigma}$ to denote a fresh symbol, called the *converse* of σ. We use notation $\Sigma^- = \{\overline{\sigma} \mid \sigma \in \Sigma^+\}$ and assume that $\Sigma^- \cap \Sigma^+ = \emptyset$. For $\varrho = \overline{\sigma} \in \Sigma^-$, we set $\overline{\varrho} \overset{\text{def}}{=} \sigma$. By an *alphabet with converse* we understand $\Sigma = \Sigma^+ \cup \Sigma^-$.

Definition 2.1. A *context-free semi-Thue system* S over Σ is a finite set of context-free production rules over alphabet Σ. We say that S is *symmetric* if, for every rule $\sigma \to \varrho_1 \dots \varrho_k$ of S, the rule $\overline{\sigma} \to \overline{\varrho}_k \dots \overline{\varrho}_1$ is also in S. $\lhd$

A context-free semi-Thue system is like a context-free grammar, but it has no designated start symbol and there is no distinction between terminal and non-terminal symbols. We assume that for $\sigma \in \Sigma$, the word σ is derivable from σ using such a grammar.

Definition 2.2. A context-free semi-Thue system S over Σ is called a *regular semi-Thue system* S over Σ if, for every $\sigma \in \Sigma$, the set of words derivable from σ using the system is a regular language over Σ. $\lhd$

Similarly as in [5], we assume that any considered regular semi-Thue system S is always given together with a mapping A that associates each $\sigma \in \Sigma$ with a finite automaton A_σ recognizing words derivable from σ using S. We call A the *mapping specifying the finite automata of* S. Note that it is undecidable to check whether a context-free semi-Thue system is regular [20].

Recall that a *finite automaton* $\mathbb{A}$ over alphabet Σ is a tuple $\langle \Sigma, Q, I, \delta, F \rangle$, where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A *run* of $\mathbb{A}$ on a word $\varrho_1 \dots \varrho_k$ is a finite sequence of states $q_0, q_1, \dots, q_k$ such that $q_0 \in I$ and $\delta(q_{i-1}, \varrho_i, q_i)$ holds for every $1 \leq i \leq k$. It is an *accepting run* if $q_k \in F$. We say that $\mathbb{A}$ *accepts* a word w if there exists an accepting run of $\mathbb{A}$ on w.

2.2 Regular Grammar Logics with Converse

Our language is based on a set Σ of *modal indices*, which is an alphabet with converse, and a set Φ_0 of *propositions.*

Definition 2.3. *Formulas* of the *base language* are defined by the following BNF grammar, where $p \in \Phi_0$ and $\sigma \in \Sigma$:

$$\varphi, \psi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \mid [\sigma]\varphi \mid \langle\sigma\rangle\varphi \qquad \lhd$$

A formula is in the *negation normal form* (NNF) if it does not contain $\to$ and uses $\neg$ only before propositions. Every formula of the base language can be transformed to an equivalent in NNF. By $\overline{\varphi}$ we denote the NNF of $\neg\varphi$.

Definition 2.4. A *Kripke model* is a tuple $M = \langle W, R, h\rangle$, where

- W is a non-empty set of *possible worlds*,
- R is a function that maps each $\sigma \in \Sigma$ to a binary relation $R_\sigma \subseteq W \times W$, called the *accessibility relation for* σ,
- h is a function that maps each $w \in W$ to a set $h(w) \subseteq \Phi_0$ of propositions that are true at w.

The accessibility relations are required to satisfy the property that, for every $\sigma \in \Sigma$, $R_{\overline{\sigma}} = R_\sigma^- \stackrel{\text{def}}{=} \{(y,x) \mid (x,y) \in R_\sigma\}$. $\lhd$

Definition 2.5. Given a Kripke model $M = \langle W, R, h\rangle$ and a world $w \in W$, the *satisfaction relation* $\models$ is defined as usual for the classical connectives with two extra clauses for the modalities as below:

$$M, w \models [\sigma]\varphi \;\text{ iff }\; \forall v \in W\; R_\sigma(w,v) \text{ implies } M, v \models \varphi,$$
$$M, w \models \langle\sigma\rangle\varphi \;\text{ iff }\; \exists v \in W\; R_\sigma(w,v) \text{ and } M, v \models \varphi.$$

We say that: φ is *satisfied* at w in M (or M *satisfies* φ at w) if $M, w \models \varphi$; M *satisfies* a set X of formulas at w, denoted by $M, w \models X$, if $M, w \models \varphi$ for all $\varphi \in X$; and M *validates* X, denoted by $M \models X$, if $M, w \models X$ for every world w of M. $\lhd$

Given two binary relations $R_1, R_2 \subseteq W \times W$, their relational composition is defined by $R_1 \circ R_2 \stackrel{\text{def}}{=} \{(x,y) \mid \exists z \in W\, (R_1(x,z) \wedge R_2(z,y))\}$.

Definition 2.6. Let S be a symmetric regular semi-Thue system over Σ. The regular grammar logic with converse corresponding to S, denoted by $L(S)$, is characterized by the class of admissible Kripke models $M = \langle W, R, h\rangle$ such that, for every rule $\sigma \to \varrho_1 \ldots \varrho_k$ of S, $R_{\varrho_1} \circ \cdots \circ R_{\varrho_k} \subseteq R_\sigma$. Such a structure is called an *L-model*, where L abbreviates $L(S)$. $\lhd$

The class of regular grammar logics with converse is denoted by REG^c.

Definition 2.7. Let L be a REG^c logic and X, Γ be finite sets of formulas. We say that X is *L-satisfiable* w.r.t. the set Γ of global assumptions if there exists an L-model that validates Γ and satisfies X at some possible world. $\lhd$

3 A Tableau Calculus for $\boldsymbol{REG^c}$

From now on, let S be a symmetric regular semi-Thue system over Σ, A be the mapping specifying the finite automata of S, and L be the REG^c logic corresponding to S. For $\sigma \in \Sigma$, we write A_σ in the form $\langle \Sigma, Q_\sigma, I_\sigma, \delta_\sigma, F_\sigma \rangle$.

For the tableau calculus defined here we extend the base language with the auxiliary modal operators $\Box_\sigma$, $[A_\sigma, q]$ and $\langle A_\sigma, q \rangle$, where $\sigma \in \Sigma$ and q is a state of A_σ. In the extended language, if φ is a formula, then $\Box_\sigma \varphi$, $[A_\sigma, q]\varphi$ and $\langle A_\sigma, q \rangle \varphi$ are also formulas. The semantics of such formulas is defined as follows.

Definition 3.1. Given a Kripke model $M = \langle W, R, h \rangle$ and a world $w \in W$, the semantics of auxiliary modalities is defined by:

- $M, w \models \Box_\sigma \varphi$ if $M, w \models [\sigma]\varphi$,
- $M, w \models [A_\sigma, q]\varphi$ (respectively $M, w \models \langle A_\sigma, q \rangle \varphi$) if $M, w_k \models \varphi$ for all (respectively some) $w_k \in W$ such that there exist worlds $w_0 = w$, w_1, ..., w_k of M, with $k \geq 0$, states $q_0 = q$, q_1, ..., q_k of A_σ with $q_k \in F_\sigma$, and a word $\varrho_1 \ldots \varrho_k$ over Σ such that $R_{\varrho_i}(w_{i-1}, w_i)$ and $\delta_\sigma(q_{i-1}, \varrho_i, q_i)$ hold for all $1 \leq i \leq k$. ◁

The operators $\Box_\sigma$ and $[A_\sigma, q]$ are universal modal operators, while $\langle A_\sigma, q \rangle$ is the existential modal operator dual to $[A_\sigma, q]$. Although $\Box_\sigma \varphi$ has the same semantics as $[\sigma]\varphi$, the operator $\Box_\sigma$ behaves differently than $[\sigma]$ in our calculus. The intuition of these auxiliary operators is as follows. Suppose that a word $\varrho_1 \ldots \varrho_n$ is derivable from σ by applying a sequence of rules of S, which may be arbitrarily long. Then $R_{\varrho_1} \circ \cdots \circ R_{\varrho_n} \subseteq R_\sigma$ holds for every L-model $\langle W, R, h \rangle$. Hence $[\sigma]\varphi \to [\varrho_1] \ldots [\varrho_n]\varphi$ is L-valid for any φ. So, having $[\sigma]\varphi$ we may need to derive $[\varrho_1] \ldots [\varrho_n]\varphi$. But n is not bounded, as the sequence of applied production rules may be arbitrarily long. The formula may then be too big. A solution to this problem depends on using the finite automaton A_σ to control the behavior of $[\sigma]$. We treat $[\sigma]\varphi$ as the conjunction of $\{[A_\sigma, q]\varphi \mid q \in I_\sigma\}$. Having $[A_\sigma, q]\varphi$ at a possible world u, if $R_\varrho(u, v)$ and $\delta_\sigma(q, \varrho, q')$ hold then we can add $[A_\sigma, q']\varphi$ to v. We deal with this by deriving $\Box_\varrho [A_\sigma, q']\varphi$ from $[A_\sigma, q]\varphi$ when $\delta_\sigma(q, \varrho, q')$ holds. We use $\Box_\varrho$ here instead of $[\varrho]$ because the modal operator is needed only for atomic ϱ-transitions and we do not need to automatize $\Box_\varrho$ as in the case of $[\varrho]$.[8]

Automaton-modal operators, especially universal ones, have previously been used, for example, in [15,18,11,16,22].

Remark 3.2. We have tried to use universal modal operators indexed by a reversed finite automaton instead of existential automaton-modal operators, but did not succeed with that. In the presence of converse, the difficulty lies in that one can travel forward and backward along the skeleton tree that unfolds the model under construction in an arbitrary way, making returns at different possible worlds and continuing the travel from the current world many times before a final return to the current world. ◁

[8] The operators $\Box_\sigma$ are introduced to simplify the rule (*cut*) given in Table 1 and make it more intuitive. The use of $\Box_\sigma$ in the rules ([A]) and ($[\text{A}]_f$) is just for convenience. The rules ([A]) and ($[\text{A}]_f$) are eliminable (by modifying the rule (*trans*) appropriately).

Definition 3.3. For a set X of formulas, by $psf(X)$ we denote the set of all formulas φ and $\overline{\varphi}$ of the base language such that either $\varphi \in X$ or φ is a subformula of some formula of X.[9] The closure $cl_L(X)$ is defined as

$$\begin{aligned} cl_L(X) = psf(X) \cup \{&[A_\sigma, q]\varphi, \Box_\varrho[A_\sigma, q]\varphi, \langle A_\sigma, q\rangle\varphi, \Box_\varrho\langle A_\sigma, q\rangle\varphi, \langle\varrho\rangle\langle A_\sigma, q\rangle\varphi \mid \\ &\sigma, \varrho \in \Sigma, q \in Q_\sigma, \varphi \in psf(X), \text{ and} \\ &([\sigma]\varphi \in psf(X) \text{ or } [A_\sigma, q']\varphi \in X \text{ for some } q')\}. \end{aligned}$$ $\triangleleft$

For $\sigma \in \Sigma$ and $q \in Q_\sigma$, we set $\delta_\sigma(q) \overset{\text{def}}{=} \{(\varrho, q') \mid (q, \varrho, q') \in \delta_\sigma\}$.

Let X and Γ be finite sets of formulas in NNF of the base language. We define now a tableau calculus $\mathcal{C}L$ for the problem of checking whether X is L-satisfiable w.r.t. the set Γ of global assumptions. We incorporate global assumptions in order to make a direct connection with description logic (DL). The set of global assumptions plays the role of a TBox of DL. It is known that in some DLs the TBox can be "internalized", but the transformation approach is not practical.

Tableau rules are written downwards, with a set of formulas above the line as the *premise* and a number of sets of formulas below the line as the *(possible) conclusions*. A tableau rule is either an *"or"-rule* or an *"and"-rule*. Possible conclusions of an "or"-rule are separated by '|', while conclusions of an "and"-rule are separated/specified using '&'. If a rule is a unary rule or an "and"-rule then its conclusions are "firm" and we ignore the word "possible". An "or"-rule has the meaning that, if the premise is L-satisfiable w.r.t. Γ then some of the possible conclusions are also L-satisfiable w.r.t. Γ. On the other hand, an "and"-rule has the meaning that, if the premise is L-satisfiable w.r.t. Γ then all of the conclusions are also L-satisfiable w.r.t. Γ (possibly at different worlds of the model under construction).

We use Y to denote a set of formulas, and Y, φ to denote the set $Y \cup \{\varphi\}$.

Definition 3.4. The *tableau calculus* $\mathcal{C}L$ w.r.t. a set Γ of global assumptions for the REG^c logic L is the set of tableau rules given in Table 1. The rule $(trans)$ is the only *"and"-rule* and the only *transitional rule*. The other rules are *"or"-rules*, which are also called *static rules*.[10] We assume that the rules $(\wedge)$, $(\vee)$, (aut), ([A]), ($[\mathrm{A}]_f$), (cut) are applicable only when the premise is a *proper* subset of each of the possible conclusions.[11] Such rules are said to be *monotonic*. $\triangleleft$

Instantiating, for example, rule $(trans)$ to $Y = \{\langle\sigma\rangle p, \langle\sigma\rangle q, \Box_\sigma r\}$ and $\Gamma = \{s\}$, we get two conclusions: $\{p, r, s\}$ and $\{q, r, s\}$.

The intuition behind distinguishing between static and transitional rules is that the static rules keep us at the same possible world of the model under construction, while each conclusion of the transitional rule takes us to a new possible world.

For any rule of $\mathcal{C}L$ except (cut) and $(trans)$, the distinguished formulas of the premise are called the *principal formulas* of the rule. The principal formulas of the rule $(trans)$ are the formulas of the form $\langle\sigma\rangle\varphi$ of the premise. The rule (cut) does not have principal formulas.

[9] Recall that $\overline{\varphi}$ is the negation normal form of φ.

[10] Unary static rules can be treated either as "and"-rules or as "or"-rules.

[11] Notice that the premise of any rule among $(\wedge)$, $(\vee)$, (aut), ([A]), ($[\mathrm{A}]_f$), (cut) is a subset of every possible conclusion of the rule.

$(\bot_0)\ \dfrac{Y, \bot}{\bot}$ $\qquad$ $(\bot)\ \dfrac{Y, p, \neg p}{\bot}$
$(\wedge)\ \dfrac{Y, \varphi \wedge \psi}{Y, \varphi \wedge \psi, \varphi, \psi}$ $\qquad$ $(\vee)\ \dfrac{Y, \varphi \vee \psi}{Y, \varphi \vee \psi, \varphi \mid Y, \varphi \vee \psi, \psi}$
$(aut)\ \dfrac{Y, [\sigma]\varphi}{Y, [\sigma]\varphi, [A_\sigma, q_1]\varphi, \ldots, [A_\sigma, q_k]\varphi}$ if $I_\sigma = \{q_1, \ldots, q_k\}$
if $\delta_\sigma(q) = \{(\varrho_1, q_1), \ldots, (\varrho_k, q_k)\}$ and $q \notin F_\sigma$: $([\mathrm{A}])\ \dfrac{Y, [A_\sigma, q]\varphi}{Y, [A_\sigma, q]\varphi, \Box_{\varrho_1}[A_\sigma, q_1]\varphi, \ldots, \Box_{\varrho_k}[A_\sigma, q_k]\varphi}$ $(\langle \mathrm{A} \rangle)\ \dfrac{Y, \langle A_\sigma, q\rangle\varphi}{Y, \langle \varrho_1\rangle\langle A_\sigma, q_1\rangle\varphi \mid \ldots \mid Y, \langle \varrho_k\rangle\langle A_\sigma, q_k\rangle\varphi}$
if $\delta_\sigma(q) = \{(\varrho_1, q_1), \ldots, (\varrho_k, q_k)\}$ and $q \in F_\sigma$: $([\mathrm{A}]_f)\ \dfrac{Y, [A_\sigma, q]\varphi}{Y, [A_\sigma, q]\varphi, \Box_{\varrho_1}[A_\sigma, q_1]\varphi, \ldots, \Box_{\varrho_k}[A_\sigma, q_k]\varphi, \varphi}$ $(\langle \mathrm{A} \rangle_f)\ \dfrac{Y, \langle A_\sigma, q\rangle\varphi}{Y, \langle \varrho_1\rangle\langle A_\sigma, q_1\rangle\varphi \mid \ldots \mid Y, \langle \varrho_k\rangle\langle A_\sigma, q_k\rangle\varphi \mid Y, \varphi}$
$(cut)\ \dfrac{Y}{Y, [A_\sigma, q]\varphi \mid Y, \Box_\varrho\langle A_\sigma, q'\rangle\overline{\varphi}}$ if Y contains a formula $\langle\varrho\rangle\psi$, $[A_\sigma, q']\varphi$ belongs to $cl_L(Y \cup \Gamma)$, and $(q', \overline{\varrho}, q) \in \delta_\sigma$
$(trans)\ \dfrac{Y}{\&\{\ (\{\varphi\} \cup \{\psi \text{ s.t. } \Box_\sigma\psi \in Y\} \cup \Gamma) \text{ s.t. } \langle\sigma\rangle\varphi \in Y\ \}}$

Table 1. Rules of the tableau calculus for REG^c

The purpose of the restriction on the applicability of the rules ($\wedge$), ($\vee$), (aut), ([A]), ([A]$_f$), (cut) is to guarantee that sequences of applications of static rules are always finite. Note that none of the static rules creates a formula of the form $\langle A_\sigma, q\rangle\varphi$ for the possible conclusions (provided that the premise is a subset of $cl_L(X \cup \Gamma)$). That is why we do not make the rules ($\langle$A$\rangle$) and ($\langle$A$\rangle_f$) monotonic. The second reason of this is that a formula of the form $\langle A_\sigma, q\rangle\varphi$ must be "reduced" as a principal formula of ($\langle$A$\rangle$) or ($\langle$A$\rangle_f$) because any one of the possible conclusions may play a key role in fulfilling the eventuality expressed by the formula.

We assume the following preferences for the rules of $\mathcal{C}L$: the rules ($\bot_0$) and ($\bot$) have the highest priority; unary static rules have a higher priority than non-unary static rules; the rule (cut) has the lowest priority among static rules; all the static rules have a higher priority than the transitional rule ($trans$).

Definition 3.5. An *"and-or" graph* for (X, Γ) w.r.t. $\mathcal{C}L$, also called a *$\mathcal{C}L$-tableau* for (X, Γ), is a rooted graph constructed as follows:

- the root of the graph has contents (i.e., is labeled by) $X \cup \Gamma$,
- for every node v of the graph, if a tableau rule of $\mathcal{C}L$ is applicable to the contents of v in the sense that an instance of the rule has the contents of v as the premise and $Z_1, \ldots, Z_k$ as the possible conclusions, then choose such a rule accordingly to the preferences[12] and apply it to v to create k successors $w_1, \ldots, w_k$ of v respectively with contents $Z_1, \ldots, Z_k$, maintaining the following constraints:
  - if the graph already contains a node w_i' with the same contents as w_i then instead of creating a new node w_i as a successor of v we just connect v to w_i' and assume $w_i = w_i'$,
  - if the applied rule is ($trans$) then we label the edge (v, w_i) by the principal formula corresponding to the successor w_i.

If the rule expanding v is an "or"-rule then v is an *"or"-node*, else v is an *"and"-node*. If no rule is applicable to v then v is an *end node*. $\lhd$

Note that each node of the graph is "expanded" only once (using one rule), and that the graph is constructed using *global caching* [26,12,14] and each of its nodes has unique contents.

Apart from monotonicity, notice also the other restrictions on the applicability of (cut). Observe that, if L is essentially a regular grammar logic without converse (in the sense that for every rule $\sigma \to \varrho_1 \ldots \varrho_k$ of S either $\{\sigma, \varrho_1, \ldots, \varrho_k\} \subseteq \Sigma^+$ or $\{\sigma, \varrho_1, \ldots, \varrho_k\} \subseteq \Sigma^-$) and the formulas of $X \cup \Gamma$ do not use modal indices from Σ^-, then the rule (cut) will never be used.

Example 3.6. Consider the regular grammar logic with converse L that corresponds to the following semi-Thue system over alphabet $\{\sigma, \varrho, \overline{\sigma}, \overline{\varrho}\}$:

$$\{\varrho \to \overline{\sigma}\varrho,\ \varrho \to \sigma,\ \overline{\varrho} \to \overline{\varrho}\sigma,\ \overline{\varrho} \to \overline{\sigma}\}.$$

The set of words derivable from ϱ is characterized by $(\overline{\sigma})^*(\sigma + \varrho)$. Let

$$A_\varrho = \langle\{\sigma, \varrho, \overline{\sigma}, \overline{\varrho}\}, \{0, 1\}, \{0\}, \{(0, \overline{\sigma}, 0), (0, \sigma, 1), (0, \varrho, 1)\}, \{1\}\rangle.$$

[12] If there are several applicable rules with the same priority, choose any one of them.

In Figures 1 and 2 we give an "and-or" graph for $(\{\langle\sigma\rangle(\varphi \lor \psi)\}, \emptyset)$ w.r.t. $\mathcal{CL}$, where $\varphi = p \land q \land [\varrho]\neg p$ and $\psi = p \land r \land [\varrho]\neg p$. The nodes are numbered when created and are expanded using DFS.[13] In each node, we display the formulas of the contents of the node, the name of rule expanding the node, and the information about whether the node is an "or"-node (when necessary). We do not display labels of edges outgoing from "and"-nodes.

Notice that:

- The rule (*cut*) is applied only once. This is due to the restrictions on the applicability of (*cut*) and the preferences of rules.
- The cache of nodes (22) and (27) is useful, as they appear on a number of different incoming paths.
- There is a cycle (22), (23), (24), (22). $\lhd$

Definition 3.7. A *marking* of an "and-or" graph G is a subgraph G' of G such that:

- the root of G is the root of G',
- if v is a node of G' and is an "or"-node of G then there exists at least one edge (v, w) of G that is an edge of G',
- if v is a node of G' and is an "and"-node of G then every edge (v, w) of G is an edge of G',
- if (v, w) is an edge of G' then v and w are nodes of G'. $\lhd$

Definition 3.8. Let G be an "and-or" graph for (X, Γ) w.r.t. $\mathcal{CL}$, G' be a marking of G, v be a node of G', and $\langle A_\sigma, q\rangle\varphi$ be a formula of the contents of v. A *trace* of $\langle A_\sigma, q\rangle\varphi$ in G' starting from v is a sequence (v_0, φ_0), …, (v_k, φ_k) such that:

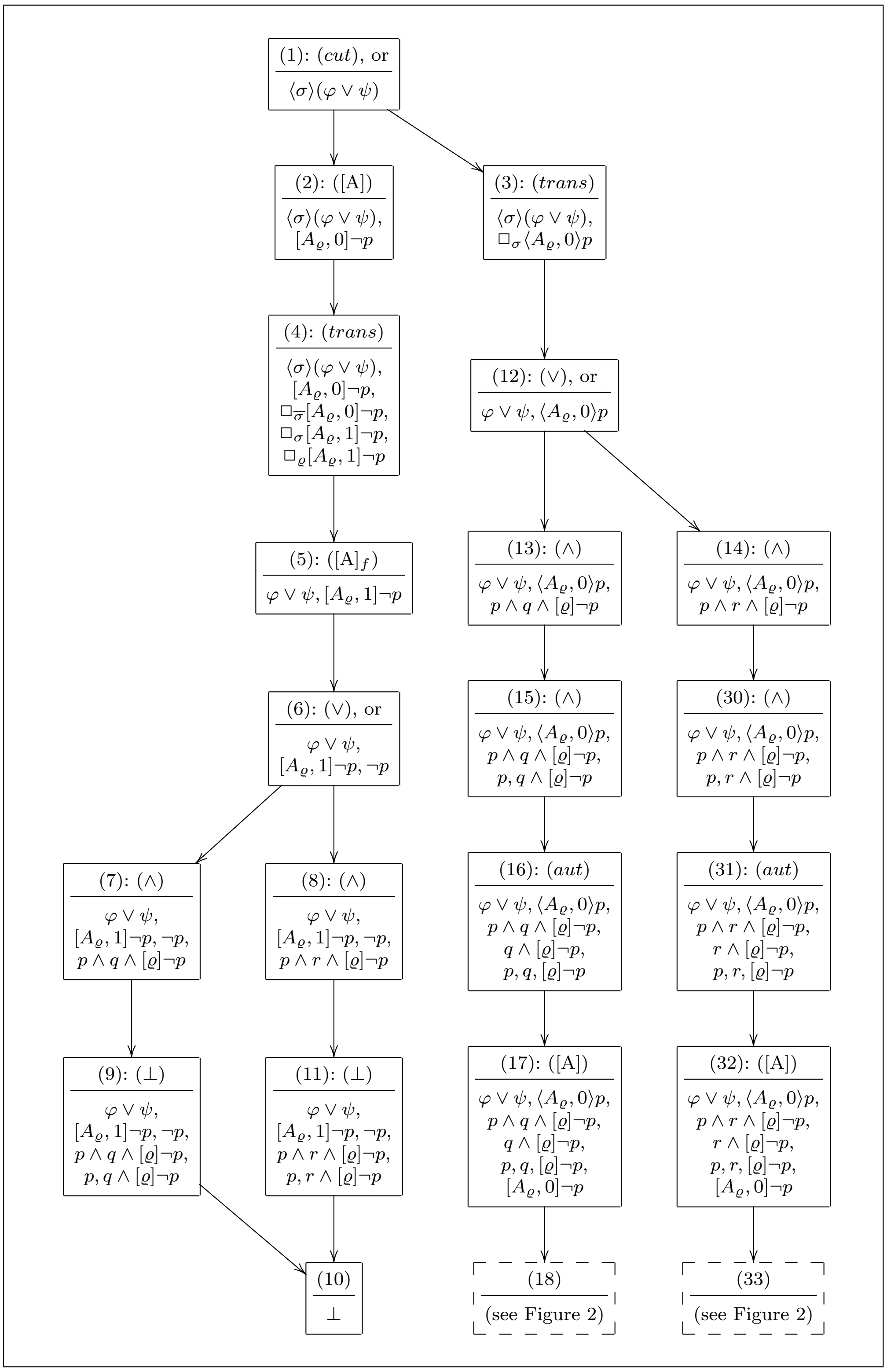
- $v_0 = v$ and $\varphi_0 = \langle A_\sigma, q\rangle\varphi$,
- for every $1 \leq i \leq k$, (v_{i-1}, v_i) is an edge of G',
- for every $1 \leq i \leq k$, φ_i is a formula of the contents of v_i such that:
  - if φ_{i-1} is not a principal formula of the tableau rule expanding v_{i-1} then the rule must be a static rule and $\varphi_i = \varphi_{i-1}$,
  - else if the rule is $(\langle\mathrm{A}\rangle)$ or $(\langle\mathrm{A}\rangle_f)$ then φ_{i-1} is of the form $\langle A_\sigma, q'\rangle\varphi$ and φ_i is the formula obtained from φ_{i-1},
  - else the rule is (*trans*), φ_{i-1} is of the form $\langle\sigma\rangle\langle A_\sigma, q'\rangle\varphi$ and is the label of the edge (v_{i-1}, v_i) and $\varphi_i = \langle A_\sigma, q'\rangle\varphi$.

A trace (v_0, φ_0), …, (v_k, φ_k) of $\langle A_\sigma, q\rangle\varphi$ in G' is called a $\Diamond$-*realization* in G' for $\langle A_\sigma, q\rangle\varphi$ at v_0 if $\varphi_k = \varphi$. $\lhd$

Definition 3.9. A marking G' of an "and-or" graph G is *consistent* if:

- *local consistency:* G' does not contain any node with contents $\{\bot\}$,
- *global consistency:* for every node v of G', every formula of the form $\langle A_\sigma, q\rangle\varphi$ of the contents of v has a $\Diamond$-realization (starting from v) in G'. $\lhd$

[13] DFS stands for the standard depth-first search algorithm for traversing graphs.



(1): (cut), or
$\langle\sigma\rangle(\varphi \lor \psi)$
(2): ([A])
$\langle\sigma\rangle(\varphi \lor \psi)$, $[A_\varrho, 0]\neg p$
(3): ($trans$)
$\langle\sigma\rangle(\varphi \lor \psi)$, $\Box_\sigma\langle A_\varrho, 0\rangle p$
(4): ($trans$)
$\langle\sigma\rangle(\varphi \lor \psi)$, $[A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(12): ($\lor$), or
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$
(5): ($[A]_f$)
$\varphi \lor \psi, [A_\varrho, 1]\neg p$
(13): ($\land$)
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land q \land [\varrho]\neg p$
(14): ($\land$)
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land r \land [\varrho]\neg p$
(6): ($\lor$), or
$\varphi \lor \psi$, $[A_\varrho, 1]\neg p, \neg p$
(15): ($\land$)
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land q \land [\varrho]\neg p$, $p, q \land [\varrho]\neg p$
(30): ($\land$)
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land r \land [\varrho]\neg p$, $p, r \land [\varrho]\neg p$
(7): ($\land$)
$\varphi \lor \psi$, $[A_\varrho, 1]\neg p, \neg p$, $p \land q \land [\varrho]\neg p$
(8): ($\land$)
$\varphi \lor \psi$, $[A_\varrho, 1]\neg p, \neg p$, $p \land r \land [\varrho]\neg p$
(16): (aut)
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land q \land [\varrho]\neg p$, $q \land [\varrho]\neg p$, $p, q, [\varrho]\neg p$
(31): (aut)
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land r \land [\varrho]\neg p$, $r \land [\varrho]\neg p$, $p, r, [\varrho]\neg p$
(9): ($\bot$)
$\varphi \lor \psi$, $[A_\varrho, 1]\neg p, \neg p$, $p \land q \land [\varrho]\neg p$, $p, q \land [\varrho]\neg p$
(11): ($\bot$)
$\varphi \lor \psi$, $[A_\varrho, 1]\neg p, \neg p$, $p \land r \land [\varrho]\neg p$, $p, r \land [\varrho]\neg p$
(17): ([A])
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land q \land [\varrho]\neg p$, $q \land [\varrho]\neg p$, $p, q, [\varrho]\neg p$, $[A_\varrho, 0]\neg p$
(32): ([A])
$\varphi \lor \psi, \langle A_\varrho, 0\rangle p$, $p \land r \land [\varrho]\neg p$, $r \land [\varrho]\neg p$, $p, r, [\varrho]\neg p$, $[A_\varrho, 0]\neg p$
(10)
$\bot$
(18)
(see Figure 2)
(33)
(see Figure 2)

Fig. 1. An example of "and-or" graph: part I

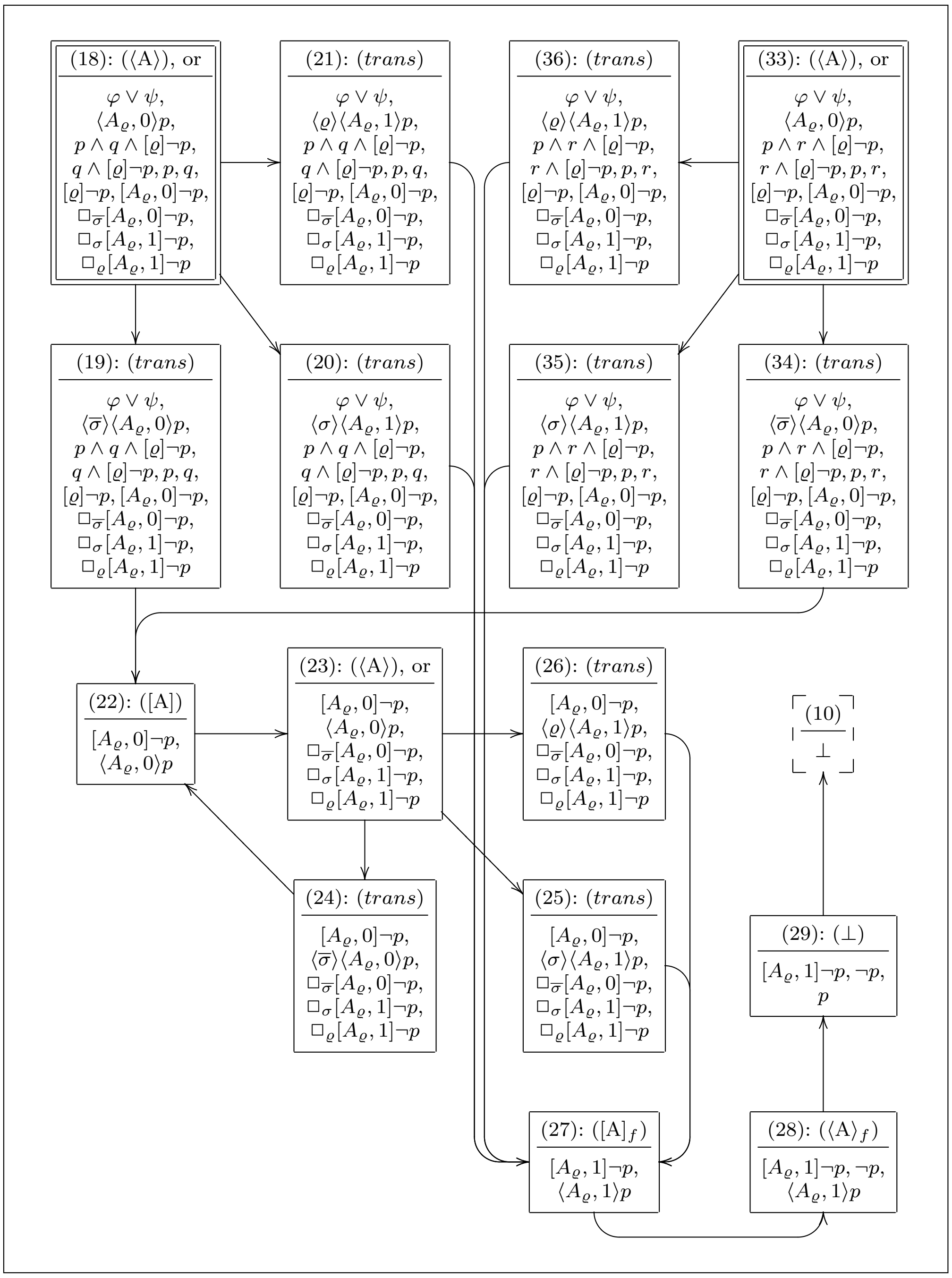

(18): (⟨A⟩), or
$\varphi \lor \psi$, $\langle A_\varrho, 0\rangle p$, $p \land q \land [\varrho]\neg p$, $q \land [\varrho]\neg p, p, q$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(21): (*trans*)
$\varphi \lor \psi$, $\langle\varrho\rangle\langle A_\varrho, 1\rangle p$, $p \land q \land [\varrho]\neg p$, $q \land [\varrho]\neg p, p, q$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(36): (*trans*)
$\varphi \lor \psi$, $\langle\varrho\rangle\langle A_\varrho, 1\rangle p$, $p \land r \land [\varrho]\neg p$, $r \land [\varrho]\neg p, p, r$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(33): (⟨A⟩), or
$\varphi \lor \psi$, $\langle A_\varrho, 0\rangle p$, $p \land r \land [\varrho]\neg p$, $r \land [\varrho]\neg p, p, r$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(19): (*trans*)
$\varphi \lor \psi$, $\langle\overline{\sigma}\rangle\langle A_\varrho, 0\rangle p$, $p \land q \land [\varrho]\neg p$, $q \land [\varrho]\neg p, p, q$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(20): (*trans*)
$\varphi \lor \psi$, $\langle\sigma\rangle\langle A_\varrho, 1\rangle p$, $p \land q \land [\varrho]\neg p$, $q \land [\varrho]\neg p, p, q$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(35): (*trans*)
$\varphi \lor \psi$, $\langle\sigma\rangle\langle A_\varrho, 1\rangle p$, $p \land r \land [\varrho]\neg p$, $r \land [\varrho]\neg p, p, r$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(34): (*trans*)
$\varphi \lor \psi$, $\langle\overline{\sigma}\rangle\langle A_\varrho, 0\rangle p$, $p \land r \land [\varrho]\neg p$, $r \land [\varrho]\neg p, p, r$, $[\varrho]\neg p, [A_\varrho, 0]\neg p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(22): ([A])
$[A_\varrho, 0]\neg p$, $\langle A_\varrho, 0\rangle p$
(23): (⟨A⟩), or
$[A_\varrho, 0]\neg p$, $\langle A_\varrho, 0\rangle p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(26): (*trans*)
$[A_\varrho, 0]\neg p$, $\langle\varrho\rangle\langle A_\varrho, 1\rangle p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(10)
$\bot$
(24): (*trans*)
$[A_\varrho, 0]\neg p$, $\langle\overline{\sigma}\rangle\langle A_\varrho, 0\rangle p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(25): (*trans*)
$[A_\varrho, 0]\neg p$, $\langle\sigma\rangle\langle A_\varrho, 1\rangle p$, $\Box_{\overline{\sigma}}[A_\varrho, 0]\neg p$, $\Box_\sigma[A_\varrho, 1]\neg p$, $\Box_\varrho[A_\varrho, 1]\neg p$
(29): (⊥)
$[A_\varrho, 1]\neg p, \neg p$, p
(27): ($[\mathrm{A}]_f$)
$[A_\varrho, 1]\neg p$, $\langle A_\varrho, 1\rangle p$
(28): ($\langle\mathrm{A}\rangle_f$)
$[A_\varrho, 1]\neg p, \neg p$, $\langle A_\varrho, 1\rangle p$

Fig. 2. An example of “and-or” graph: part II

Theorem 3.10 (Soundness and Completeness of $\mathcal{C}L$). *Let S be a symmetric regular semi-Thue system over Σ, A be the mapping specifying the finite automata of S, and L be the REG^c logic corresponding to S. Let X and Γ be finite sets of formulas in NNF of the base language, and G be an "and-or" graph for (X, Γ) w.r.t. $\mathcal{C}L$. Then X is L-satisfiable w.r.t. the set Γ of global assumptions iff G has a consistent marking.* $\lhd$

The "only if" direction means soundness of $\mathcal{C}L$, while the "if" direction means completeness of $\mathcal{C}L$. This theorem follows from Lemmas 4.1 and 4.11, which are given and proved in the next section.

Reconsider Example 3.6. The "and-or" graph given in Figures 1 and 2 does not have any consistent marking. The graph contains some markings (e.g., the one consisting of nodes (1), (3), (12), (13), (15)-(19), (22)-(24)) that satisfy the local consistency property, but these markings do not satisfy the global consistency property because the formula $\langle A_\varrho, 0\rangle p$ of (22) does not have any $\Diamond$-realization in the mentioned markings. By Theorem 3.10, the formula $\langle\sigma\rangle((p \land q \land [\varrho]\neg p) \lor (p \land r \land [\varrho]\neg p))$ is unsatisfiable (w.r.t. $\emptyset$) in the REG^c logic specified in the example.

Observe that if we generalize the rule $(\bot)$ to "derive $\bot$ from φ and $\overline{\varphi}$", which is supported by Lemma 4.8 and the proof of Lemma 4.11, then the nodes in Figure 2 of the mentioned example can be discarded by connecting the nodes (17) and (32) to (10). Furthermore, if unary static rules are implicitly applied by "normalizing" formulas then the graph can significantly be further simplified.

4 Proofs of Soundness and Completeness

4.1 Soundness

Lemma 4.1. *Let S, A, L, X, Γ, G be as described in Theorem 3.10. Suppose that X is L-satisfiable w.r.t. the set Γ of global assumptions. Then G has a consistent marking.*

Proof. We construct a consistent marking G' of G as follows. At the beginning, G' contains only the root of G. Then, for every node v of G' and for every successor w of v in G, if the contents of w are L-satisfiable w.r.t. Γ, then add the node w and the edge (v, w) to G'.

To prove that G' is a marking of G we need to show that:

1. for every "or"-rule of $\mathcal{C}L$, if the premise is L-satisfiable w.r.t. Γ then one of the possible conclusions of the rule is also L-satisfiable w.r.t. Γ,
2. for every "and"-rule of $\mathcal{C}L$, if the premise is L-satisfiable w.r.t. Γ then so is each conclusion of the rule.

We consider here only the rule (cut) and leave the others to the reader. Suppose that Y is L-satisfiable w.r.t. Γ and let $M = \langle W, R, h\rangle$ be an L-model that validates Γ and satisfies Y at a possible world u. Suppose that $\delta_\sigma(q', \overline{\varrho}, q)$ holds and $M, u \not\models \Box_\varrho\langle A_\sigma, q'\rangle\overline{\varphi}$. We show that $M, u \models [A_\sigma, q]\varphi$. We have that $M, u \models \langle\varrho\rangle[A_\sigma, q']\varphi$. Hence there exists u' such that $R_\varrho(u, u')$ holds and $M, u' \models [A_\sigma, q']\varphi$. Since $R_{\overline{\varrho}}(u', u)$ and $\delta_\sigma(q', \overline{\varrho}, q)$ hold, it follows that $M, u \models [A_\sigma, q]\varphi$.

Clearly, G' satisfies the local consistency property.

We now check the global consistency property of G'. Let v_0 be a node of G', Y be the contents of v_0, and $\langle A_\sigma, q\rangle\varphi$ be a formula of Y. We show that the formula has a $\Diamond$-realization in G'. As Y is L-satisfiable w.r.t. Γ, there exists an L-model M that validates Γ and satisfies Y at a world u. Since $M, u \models \langle A_\sigma, q\rangle\varphi$,

> there exist worlds $u_0 = u$, u_1, …, u_k of M, with $k \geq 0$, states $q_0 = q$, $q_1, \ldots, q_k$ of A_σ with $q_k \in F_\sigma$, and a word $\varrho_1 \ldots \varrho_k$ over Σ such that $M, u_k \models \varphi$, $R_{\varrho_i}(u_{i-1}, u_i)$ and $\delta_\sigma(q_{i-1}, \varrho_i, q_i)$ hold for all $1 \leq i \leq k$. (1)

We construct a $\Diamond$-realization $(v_0, \varphi_0), \ldots, (v_h, \varphi_h)$ in G' for $\langle A_\sigma, q\rangle\varphi$ at v_0 and a map $f : \{v_0, \ldots, v_h\} \to \{u_0, \ldots, u_k\}$ such that $f(v_0) = u_0$, $f(v_h) = u_k$, and for every $0 \leq i < h$, if $f(v_i) = u_j$ then $f(v_{i+1})$ is either u_j or u_{j+1}. We maintain the following invariants for $0 \leq i \leq h$:

– the chain $(v_0, \varphi_0), \ldots, (v_i, \varphi_i)$ is a trace of $\langle A_\sigma, q\rangle\varphi$ in G' (2)

– the contents of v_i are satisfied at the world $f(v_i)$ of M (3)

– if $f(v_i) = u_j$ and $j < k$ then either $\varphi_i = \langle A_\sigma, q_j\rangle\varphi$ or $\varphi_i = \langle \varrho_{j+1}\rangle\langle A_\sigma, q_{j+1}\rangle\varphi$ (4)

– if $f(v_i) = u_k$ then either $\varphi_i = \langle A_\sigma, q_k\rangle\varphi$ or $\varphi_i = \varphi$. (5)

With $\varphi_0 = \langle A_\sigma, q_0\rangle\varphi$ and $f(v_0) = u_0$, the invariants clearly hold for $i = 0$.

Set $i := 0$. While $\varphi_i \neq \varphi$ do:

- Case v_i is expanded using a static rule but φ_i is not the principal formula:
  Let v_{i+1} be the successor of v_i such that (v_i, v_{i+1}) is an edge of G' and the contents of v_{i+1} are satisfied at the world $f(v_i)$ of M. Such a node v_{i+1} exists because the contents of v_i are satisfied at the world $f(v_i)$ of M. Let $\varphi_{i+1} = \varphi_i$, $f(v_{i+1}) = f(v_i)$, and set $i := i + 1$. Clearly, the invariants still hold.
- Case v_i is expanded using a static rule and φ_i is the principal formula:
  Let $f(v_i) = u_j$. We now have two cases:
  - Case $j < k$: Since the applied rule is a static rule, by the invariant (4), we must have $\varphi_i = \langle A_\sigma, q_j\rangle\varphi$, and the applied rule is either $(\langle\mathrm{A}\rangle)$ or $(\langle\mathrm{A}\rangle_f)$. Let $\varphi_{i+1} = \langle \varrho_{j+1}\rangle\langle A_\sigma, q_{j+1}\rangle\varphi$ and let v_{i+1} be the successor of v_i with φ_{i+1} replacing φ_i. By (1), φ_{i+1} is satisfied at u_j in M, and hence, by the invariant (3), the contents of v_{i+1} are satisfied at u_j in M. Let $f(v_{i+1}) = u_j$ and set $i := i + 1$. Clearly, the invariants still hold.
  - Case $j = k$: Since the applied rule is a static rule and $\varphi_i \neq \varphi$, by the invariant (5), we have that $\varphi_i = \langle A_\sigma, q_k\rangle\varphi$, and the applied rule is $(\langle\mathrm{A}\rangle_f)$. Let $\varphi_{i+1} = \varphi$ and let v_{i+1} be the successor of v_i with φ_{i+1} replacing φ_i. By (1), φ_{i+1} is satisfied at u_k in M. Let $f(v_{i+1}) = u_k$ and set $i := i + 1$. Clearly, the invariants still hold.
- Case v_i is expanded using the transitional rule:
  Let $f(v_i) = u_j$. Since the applied rule is the transitional rule and $\varphi_i \neq \varphi$, by the invariants (4) and (5), $\varphi_i = \langle \varrho_{j+1}\rangle\langle A_\sigma, q_{j+1}\rangle\varphi$. Let (v_i, v_{i+1}) be the edge of G with the label φ_i. Let $\varphi_{i+1} = \langle A_\sigma, q_{j+1}\rangle\varphi$ and $f(v_{i+1}) = u_{j+1}$. Clearly, the invariant (2) holds for $i + 1$. By (1), φ_{i+1} is satisfied at the world u_{j+1} of M. By the invariant (3), the other formulas of the contents of v_{i+1} are also satisfied at the world u_{j+1} of M. That is, the invariant (3) holds for $i + 1$. Clearly, the invariants (4) and (5) remain

true after increasing i by 1. So, by setting $i := i + 1$, all the invariants (2)- (5) still hold.

It remains to show that the loop terminates. Observe that the length of any sequence of applications of static rules that contribute to the trace $(v_0, \varphi_0), \ldots, (v_i, \varphi_i)$ of $\langle A_\sigma, q\rangle\varphi$ in G' is finitely bounded. That is, sooner or later either $\varphi_i = \varphi$ or v_i is a node that is expanded by the transitional rule. In the second case, if $f(v_i) = u_j$ then $f(v_{i+1}) = u_{j+1}$. As the image of f is $\{u_0, \ldots, u_k\}$, the construction of the trace must end at some step (with $\varphi_i = \varphi$) and we obtain a $\Diamond$-realization in G' for $\langle A_\sigma, q\rangle\varphi$ at v_0. This completes the proof. $\lhd$

4.2 Model Graphs

We will prove completeness of $\mathcal{CL}$ via model graphs. The technique has been used in [27,10,21] for logics without induction rules (like the one of PDL).

Definition 4.2. A *model graph* is a tuple $\langle W, R, H\rangle$, where W is a set of nodes, R is a mapping that maps each $\sigma \in \Sigma$ to a binary relation R_σ on W, and H is a function that maps each node of W to a set of formulas. $\lhd$

We use model graphs merely as data structures, but we are interested in "consistent" and "saturated" model graphs defined below. Model graphs differ from "and-or" graphs in that a model graph contains only "and"-nodes and its edges are labeled by accessibility relations. Roughly speaking, given an "and-or" graph G with a consistent marking G', to construct a model graph one can stick together the nodes in a "saturation path" of a node of G' to create a node for the model graph. Details will be given later.

A trace of a formula $\langle A_\sigma, q\rangle\varphi$ at a node in a model graph is defined analogously as for the case of "and-or" graphs:

Definition 4.3. Given a model graph $M = \langle W, R, H\rangle$ and a node $v \in W$, a *trace* of a formula $\langle A_\sigma, q\rangle\varphi \in H(v)$ (starting from v) is a chain (v_0, φ_0), …, (v_k, φ_k) such that:

- $v_0 = v$ and $\varphi_0 = \langle A_\sigma, q\rangle\varphi$,
- for every $1 \le i \le k$, $\varphi_i \in H(v_i)$,
- for every $1 \le i \le k$, if $v_i = v_{i-1}$ then:
  - φ_{i-1} is of the form $\langle A_\sigma, q'\rangle\varphi$, and
  - either $\varphi_i = \langle\varrho\rangle\langle A_\sigma, q''\rangle\varphi$ for some ϱ and q'' such that $\delta_\sigma(q', \varrho, q'')$
  - or $\varphi_i = \varphi$ and $q' \in F_\sigma$ and $i = k$,
- for every $1 \le i \le k$, if $v_i \ne v_{i-1}$ then:
  - φ_{i-1} is of the form $\langle\varrho\rangle\langle A_\sigma, q'\rangle\varphi$ and $\varphi_i = \langle A_\sigma, q'\rangle\varphi$ and $(v_{i-1}, v_i) \in R_\varrho$. $\lhd$

Definition 4.4. A trace (v_0, φ_0), …, (v_k, φ_k) of $\langle A_\sigma, q\rangle\varphi$ in a model graph M is called a $\Diamond$*-realization* for $\langle A_\sigma, q\rangle\varphi$ at v_0 if $\varphi_k = \varphi$. $\lhd$

Similarly as for markings of "and-or" graphs, we define that:

Definition 4.5. A model graph $M = \langle W, R, H\rangle$ is *consistent* if:

- *local consistency:* for every $v \in W$, $H(v)$ contains neither $\bot$ nor a *clashing pair* of the form p, $\neg p$;

– *global consistency:* for every $v \in W$, every formula $\langle A_\sigma, q\rangle\varphi$ of $H(v)$ has a $\Diamond$-realization (at v). $\lhd$

Definition 4.6. A model graph $M = \langle W, R, H\rangle$ is said to be *$\mathcal{C}L$-saturated* if the following conditions hold for every $v \in W$:

– for every $\varphi \in H(v)$:
  - if $\varphi = \psi \wedge \xi$ then $\{\psi, \xi\} \subset H(v)$,
  - if $\varphi = \psi \vee \xi$ then $\psi \in H(v)$ or $\xi \in H(v)$,
  - if $\varphi = \langle\sigma\rangle\psi$ then there exists w such that $R_\sigma(v, w)$ and $\psi \in H(w)$,
  - if $\varphi = [\sigma]\psi$ and $I_\sigma = \{q_1, \ldots, q_k\}$ then $\{[A_\sigma, q_1]\psi, \ldots, [A_\sigma, q_k]\psi\} \subset H(v)$,
  - if $\varphi = [A_\sigma, q]\psi$ and $\delta_\sigma(q) = \{(\varrho_1, q_1), \ldots, (\varrho_k, q_k)\}$ then $\{\Box_{\varrho_1}[A_\sigma, q_1]\psi, \ldots, \Box_{\varrho_k}[A_\sigma, q_k]\psi\} \subset H(v)$,
  - if $\varphi = [A_\sigma, q]\psi$ and $q \in F_\sigma$ then $\psi \in H(v)$,
  - if $\varphi = \Box_\sigma\psi$ and $R_\sigma(v, w)$ holds then $\psi \in H(w)$,

– if $R_\varrho(v, w)$ holds and $[A_\sigma, q']\varphi \in H(w)$ and $(q', \overline{\varrho}, q) \in \delta_\sigma$ then $[A_\sigma, q]\varphi \in H(v)$ or $\Box_\varrho\langle A_\sigma, q'\rangle\overline{\varphi} \in H(v)$. $\lhd$

The last condition of the above definition corresponds to the rule (*cut*). As shown in the proof of Lemma 4.9, it can be strengthened to

"if $R_\varrho(v, w)$ holds and $[A_\sigma, q']\varphi \in H(w)$ and $(q', \overline{\varrho}, q) \in \delta_\sigma$ then $[A_\sigma, q]\varphi \in H(v)$".

Definition 4.7. Given a model graph $M = \langle W, R, H\rangle$, the *L-model corresponding to M* is the Kripke model $M' = \langle W, R', h\rangle$ such that:

– $h(w) = \{p \in \Phi_0 \mid p \in H(w)\}$ for $w \in W$, and
– R'_σ for $\sigma \in \Sigma$ are the smallest binary relations on W such that:
  - $R_\sigma \subseteq R'_\sigma$ and $R'_{\overline{\sigma}} = (R'_\sigma)^-$ for every $\sigma \in \Sigma$, and
  - if $\sigma \to \varrho_1 \ldots \varrho_k \in S$, where S is the symmetric regular semi-Thue system of L, then $R'_{\varrho_1} \circ \cdots \circ R'_{\varrho_k} \subseteq R'_\sigma$. $\lhd$

Define the NNF of $\neg\Box_\sigma\varphi$ to be $\langle\sigma\rangle\overline{\varphi}$. Recall that the NNF of $\neg[\sigma]\varphi$, $\neg\langle\sigma\rangle\varphi$, $\neg[A_\sigma, q]\varphi$, $\neg\langle A_\sigma, q\rangle\varphi$ are $\langle\sigma\rangle\overline{\varphi}$, $[\sigma]\overline{\varphi}$, $\langle A_\sigma, q\rangle\overline{\varphi}$, $[A_\sigma, q]\overline{\varphi}$, respectively.

Lemma 4.8. *Let Γ be a finite set of formulas in NNF of the base language and $M = \langle W, R, H\rangle$ be a consistent and $\mathcal{C}L$-saturated model graph. Then, for any $w \in W$, $H(w)$ does not contain both φ and $\overline{\varphi}$.*

Proof. By induction on the structure of φ, using the global consistency. $\lhd$

Lemma 4.9. *Let X and Γ be finite sets of formulas in NNF of the base language, and let $M = \langle W, R, H\rangle$ be a consistent and $\mathcal{C}L$-saturated model graph such that $\Gamma \subseteq H(w)$ for all $w \in W$ and $X \subseteq H(\tau)$ for some $\tau \in W$. Then the L-model M' corresponding to M validates Γ and satisfies X at τ.*

Proof. We first show the following two assertions:

– if $[A_\sigma, q]\psi \in H(w)$ and $R_\varrho(w, w')$ and $\delta_\sigma(q, \varrho, q')$ then $[A_\sigma, q']\psi \in H(w')$ (6)
– if $[A_\sigma, q]\psi \in H(w)$ and $R_{\overline{\varrho}}(w', w)$ and $\delta_\sigma(q, \varrho, q')$ then $[A_\sigma, q']\psi \in H(w')$. (7)

Assertion (6) holds because $[A_\sigma, q]\psi \in H(w)$ and $\delta_\sigma(q, \varrho, q')$ imply $\Box_\varrho[A_\sigma, q']\psi \in H(w)$, which together with $R_\varrho(w, w')$ implies $[A_\sigma, q']\psi \in H(w')$.

For assertion (7), suppose that $[A_\sigma, q]\psi \in H(w)$ and $R_{\overline{\varrho}}(w', w)$ and $\delta_\sigma(q, \varrho, q')$ hold. Since M is $\mathcal{CL}$-saturated, either $[A_\sigma, q']\psi \in H(w')$ or $\Box_{\overline{\varrho}}\langle A_\sigma, q\rangle\overline{\psi} \in H(w')$. If $\Box_{\overline{\varrho}}\langle A_\sigma, q\rangle\overline{\psi} \in H(w')$, then $\langle A_\sigma, q\rangle\overline{\psi} \in H(w)$ (since $R_{\overline{\varrho}}(w', w)$ holds), which, by Lemma 4.8, contradicts the fact that $[A_\sigma, q]\psi \in H(w)$. Therefore $[A_\sigma, q']\psi \in H(w')$.

Let $M' = \langle W, R', h\rangle$. We now prove our lemma by induction on the construction of φ that if $\varphi \in H(u)$ for an arbitrary $u \in W$ and φ is not of the form $\Box_\sigma\psi$ nor $[A_\sigma, q]\psi$ then $M', u \models \varphi$. It suffices to consider only the non-trivial case when φ is of the form $[\sigma]\psi$. Suppose that $\varphi = [\sigma]\psi$ and $\varphi \in H(u)$. Let $v \in W$ be a world of M' such that $R'_\sigma(u, v)$ holds. We show that $M', v \models \psi$.

Since $R'_\sigma(u, v)$ holds, by the definition of M', there exist elements $w_0, \ldots, w_k$ of W and a word $\varrho_1 \ldots \varrho_k$ accepted by A_σ such that $w_0 = u$, $w_k = v$, and for every $1 \le i \le k$, either $R_{\varrho_i}(w_{i-1}, w_i)$ or $R_{\overline{\varrho}_i}(w_i, w_{i-1})$ holds. Let $q_0, \ldots, q_k$ be an accepting run of A_σ on the word $\varrho_1 \ldots \varrho_k$. We have that $q_0 \in I_\sigma$ and $q_k \in F_\sigma$. Since $[\sigma]\psi \in H(w_0)$, we also have that $[A_\sigma, q_0]\psi \in H(w_0)$. For $1 \le i \le k$, since $[A_\sigma, q_{i-1}]\psi \in H(w_{i-1})$ and $R_{\varrho_i}(w_{i-1}, w_i) \vee R_{\overline{\varrho}_i}(w_i, w_{i-1})$ and $\delta_\sigma(q_{i-1}, \varrho_i, q_i)$ hold, by assertions (6) and (7), we have that $[A_\sigma, q_i]\psi \in H(w_i)$. Thus $[A_\sigma, q_k]\psi \in H(w_k)$. Since $q_k \in F_\sigma$ and $w_k = v$, it follows that $\psi \in H(v)$. By the inductive assumption, it follows that $M', v \models \psi$, which completes the proof. ◁

4.3 Completeness

Definition 4.10. Let G' be a consistent marking of an "and-or" graph and let v be a node of G'. A *saturation path* of v w.r.t. G' is a finite sequence $v_0 = v$, v_1, …, v_k of nodes of G', with $k \ge 0$, such that, for every $0 \le i < k$, v_i is an "or"-node and (v_i, v_{i+1}) is an edge of G', and v_k is an "and"-node. ◁

Observe that there always exists a saturation path of v w.r.t. G'.

Lemma 4.11. *Let X and Γ be finite sets of formulas in NNF of the base language, and G be an "and-or" graph for (X, Γ) w.r.t. $\mathcal{CL}$. Suppose that G has a consistent marking G'. Then X is L-satisfiable w.r.t. the set Γ of global assumptions.*

Proof. We construct a model graph $M = \langle W, R, H\rangle$ as follows:

1. Let v_0 be the root of G' and $v_0, \ldots, v_k$ be a saturation path of v_0 w.r.t. G'. Set $R_\sigma = \emptyset$ for all $\sigma \in \Sigma$ and set $W = \{\tau\}$, where τ is a new node. Set $H(\tau)$ to the sum of the contents of all $v_0, \ldots, v_k$. Mark τ as *unresolved* and set $f(\tau) = v_k$. (Each node of M will be marked either as unresolved or as resolved, and f will map each node of M to an "and"-node of G'.)
2. While W contains unresolved nodes, take one unresolved node w_0 and do:
   (a) For every formula $\langle\varrho\rangle\varphi \in H(w_0)$ do:

i. Let $\varphi_0 = \langle\varrho\rangle\varphi$ and $\varphi_1 = \varphi$.
ii. Let $u_0 = f(w_0)$ and let u_1 be the node of G' such that the edge (u_0, u_1) is labeled by φ_0. (As a maintained property of f, φ_0 belongs to the contents of u_0, and hence φ_1 belongs to the contents of u_1.)
iii. If φ is of the form $\langle A_\sigma, q\rangle\psi$ then:
   A. Let $(u_1, \varphi_1), \ldots, (u_l, \varphi_l)$ be a $\Diamond$-realization in G' for φ_1 at u_1.
   B. Let $u_l, \ldots, u_m$ be a saturation path of u_l w.r.t. G'.
iv. Else let $u_1, \ldots, u_m$ be a saturation path of u_1 w.r.t. G'.
v. Let $j_0 = 0 < j_1 < \ldots < j_{n-1} < j_n = m$ be all the indices such that, for $0 \leq j \leq m$, u_j is an "and"-node of G iff $j \in \{j_0, \ldots, j_n\}$. For $0 \leq s \leq n-1$, let $\langle\varrho_s\rangle\varphi_{j_s+1}$ be the label of the edge (u_{j_s}, u_{j_s+1}) of G'. (We have that $\varrho_0 = \varrho$.)
vi. For $1 \leq s \leq n$ do:
   A. Let Z_s be the sum of the contents of the nodes $u_{j_{s-1}+1}, \ldots, u_{j_s}$.
   B. If there does not exist $w_s \in W$ such that $H(w_s) = Z_s$ then: add a new node w_s to W, set $H(w_s) = Z_s$, mark w_s as unresolved, and set $f(w_s) = u_{j_s}$.
   C. Add the pair (w_{s-1}, w_s) to $R_{\varrho_{s-1}}$.

(b) Mark w_0 as resolved.

As H is a one-to-one function and $H(w)$ of each $w \in W$ is a subset of the closure $cl_L(X \cup \Gamma)$, the above construction terminates and results in a finite model graph.

Observe that, in the above construction we transform the chain $u_0, \ldots, u_m$ of nodes of G' to a chain $w_0, \ldots, w_n$ of nodes of M by sticking together nodes in every maximal saturation path. Hence, M is $\mathcal{C}L$-saturated and satisfies the local consistency property. For $w'_0 \in W$ and $\langle A_\sigma, q'\rangle\psi \in H(w'_0)$, the formula has a trace of length 2, whose second pair is either (w'_0, ψ) or $(w_0, \langle\varrho\rangle\langle A_\sigma, q\rangle\psi)$ for some w_0, ϱ, q. This together with Step 2(a)iiiA implies that M satisfies the global consistency property. Hence, M is a consistent and $\mathcal{C}L$-saturated model graph.

Consider Step 1 of the construction. As the contents of v_0 are $X \cup \Gamma$, we have that $X \subseteq H(\tau)$ and $\Gamma \subseteq H(\tau)$. Consider Step 2(a)vi of the construction, as $u_{j_{s-1}}$ is an "and"-node and $u_{j_{s-1}+1}$ is a successor of $u_{j_{s-1}}$ that is created by the transitional rule, the contents of $u_{j_{s-1}+1}$ contain Γ. Hence $\Gamma \subseteq H(w_s)$ for every $w_s \in W$. By Lemma 4.9, the Kripke model corresponding to M validates Γ and satisfies X at τ. Hence, X is L-satisfiable w.r.t. Γ. $\triangleleft$

5 An ExpTime Tableau Decision Procedure for $\boldsymbol{REG^c}$

In this section, we present a simple ExpTime tableau algorithm for checking L-satisfiability of a given set X of formulas w.r.t. a given set Γ of global assumptions. We also briefly discuss optimizations for the algorithm.

Define the *length* of a formula φ to be the number of symbols occurring in φ. For example, the length of $\langle A_\sigma, q\rangle\psi$ is the length of ψ plus 5, treating A_σ as a symbol. Define the *size* of a finite set of formulas to be the length of the conjunction of its formulas. Define the size of a finite automaton $\langle\Sigma, Q, I, \delta, F\rangle$ to be $|Q| + |I| + |\delta| + |F|$, where $|\cdot|$ denotes the cardinality of the set.

5.1 The Basic Algorithm

Let X and Γ be finite sets of formulas in NNF of the base language, G be an "and-or" graph for (X, Γ) w.r.t. $\mathcal{C}L$, and G' be a marking of G.

Definition 5.1. The *graph G_t of traces of G' in G* is defined as follows:

- nodes of G_t are pairs (v, φ), where v is a node of G and φ is a formula of the contents of v,
- a pair $((v, \varphi), (w, \psi))$ is an edge of G_t if v is a node of G', φ is of the form $\langle A_\sigma, q\rangle\xi$ or $\langle\varrho\rangle\langle A_\sigma, q\rangle\xi$, and the sequence (v, φ), (w, ψ) is a fragment of a trace in G'.

A node (v, φ) of G_t is an *end node* if φ is a formula of the base language. A node of G_t is *productive* if there is a path connecting it to an end node. $\lhd$

In Figure 3 we present Algorithm 1 for checking L-satisfiability of X w.r.t. Γ. The algorithm starts by constructing an "and-or" graph G, with root v_0, for (X, Γ) w.r.t. $\mathcal{C}L$. After that it collects the nodes of G whose contents are L-unsatisfiable w.r.t. Γ. Such nodes are said to be *unsat* and kept in the set $UnsatNodes$. Initially, if G contains a node with contents $\{\bot\}$ then the node is *unsat*. When a node or a number of nodes become *unsat*, the algorithm propagates the status *unsat* backwards through the "and-or" graph using the procedure *updateUnsatNodes* presented in Figure 3. This procedure has the property that, after its execution, if the root v_0 of G does not belong to $UnsatNodes$ then the maximal subgraph of G without nodes from $UnsatNodes$, denoted by G', is a marking of G. After each calling of *updateUnsatNodes*, the algorithm finds the nodes of G' that make the marking not satisfying the global consistency property. Such a task is done by creating the graph G_t of traces of G' in G and finding nodes v of G' such that the contents of v contain a formula of the form $\langle A_\sigma, q\rangle\varphi$ but $(v, \langle A_\sigma, q\rangle\varphi)$ is not a productive node of G_t. If the set V of such nodes is empty then G' is a consistent marking (provided that $v_0 \notin UnsatNodes$) and the algorithm stops with a positive answer. Otherwise, V is used to update $UnsatNodes$ by calling *updateUnsatNodes*$(G, UnsatNodes, V)$. After that call, if $v_0 \in UnsatNodes$ then the algorithm stops with a negative answer, else the algorithm repeats the loop of collecting *unsat* nodes. Note that we can construct G_t only the first time and update it appropriately each time when $UnsatNodes$ is changed.

Lemma 5.2. *Let*

- *S be a symmetric regular semi-Thue system over Σ,*
- *A be the mapping specifying the finite automata of S,*
- *L be the REG^c logic corresponding to S,*
- *X and Γ be finite sets of formulas in NNF of the base language,*
- *G be an "and-or" graph for (X, Γ) w.r.t. $\mathcal{C}L$,*
- *$l = |\Sigma|$,*
- *m be the sum of the sizes of the automata A_σ for $\sigma \in \Sigma$,*
- *n be the size of $X \cup \Gamma$.*

Then G has $2^{O(l \times m \times n)}$ nodes and the contents of each node of G has $O(l \times m \times n)$ formulas and is of size $O(l \times m \times n^2)$.

Algorithm 1
Input: finite sets X and $\varGamma$ of formulas in NNF of the base language,
the mapping A specifying the finite automata of the symmetric
regular semi-Thue system of the considered REG^c logic L.
Output: *true* if X is L-satisfiable w.r.t. $\varGamma$, and *false* otherwise.

```
1. construct an "and-or" graph $G$, with root $v_0$, for $(X, \varGamma)$ w.r.t. $\mathcal{C}L$
2. $UnsatNodes := \emptyset$
3. if $G$ contains a node $v$ with contents $\{\bot\}$ then
     $updateUnsatNodes(G, UnsatNodes, \{v\})$
4. if $v_0 \in UnsatNodes$ then return false
5. let $G'$ be the maximal subgraph of $G$ without nodes from $UnsatNodes$
   (we have that $G'$ is a marking of $G$)
6. construct the graph $G_t$ of traces of $G'$ in $G$
7. while $v_0 \notin UnsatNodes$ do:
   (a) let $V$ be the set of all nodes $v$ of $G'$ such that the contents of $v$ contain a formula
       of the form $\langle A_\sigma, q\rangle\varphi$ but $(v, \langle A_\sigma, q\rangle\varphi)$ is not a productive node of $G_t$
   (b) if $V = \emptyset$ then return true
   (c) $updateUnsatNodes(G, UnsatNodes, V)$
   (d) if $v_0 \in UnsatNodes$ then return false
   (e) let $G'$ be the maximal subgraph of $G$ without nodes from $UnsatNodes$
       (we have that $G'$ is a marking of $G$)
   (f) update $G_t$ to the graph of traces of $G'$ in $G$
```

Procedure $updateUnsatNodes(G, UnsatNodes, V)$
Input: an "and-or" graph G and sets $UnsatNodes$, V of nodes of G,
where V contains new *unsat* nodes.
Output: a new set $UnsatNodes$.

```
1. $UnsatNodes := UnsatNodes \cup V$
2. while $V$ is not empty do:
   (a) remove a node $v$ from $V$
   (b) for every father node $u$ of $v$, if $u \notin UnsatNodes$ and either $u$ is an "and"-node or $u$
       is an "or"-node and all the successor nodes of $u$ belong to $UnsatNodes$ then add $u$
       to both $UnsatNodes$ and $V$
```

Fig. 3. An algorithm for checking L-satisfiability of X w.r.t. $\varGamma$

Proof. Note that $psf(X \cup \varGamma)$ has $O(n)$ formulas and $cl_L(X \cup \varGamma)$ has $O(l \times m \times n)$ formulas. Since the contents of each node of G are a subset of $cl_L(X \cup \varGamma)$, it has $O(l \times m \times n)$ formulas and is of size $O(l \times m \times n^2)$. Since the contents of the nodes of G are unique, G has $2^{O(l \times m \times n)}$ nodes. $\lhd$

Lemma 5.3. *Let S, A, L, X, $\varGamma$, l, m, n be as in Lemma 5.2. Then the execution of Algorithm 1 for X, $\varGamma$, A runs in $2^{O(l \times m \times n)}$ steps.*

Proof. By Lemma 5.2, the graph G can be constructed in $2^{O(l \times m \times n)}$ steps and has $2^{O(l \times m \times n)}$ nodes. As the contents of each node of G contain $O(l \times m \times n)$ formulas, each time when $UnsatNodes$ is extended G_t can be constructed or updated in $2^{O(l \times m \times n)}$ steps. Computing the set V can be done in polynomial time in the size of G_t, and hence also in $2^{O(l \times m \times n)}$ steps. An execution of *updateUnsatNodes* is done in polynomial time in the size of G, and hence also in $2^{O(l \times m \times n)}$ steps. As the set $UnsatNodes$ is extended at most $2^{O(l \times m \times n)}$ times, the total time for executing Algorithm 1 is $2^{O(l \times m \times n)}$. $\lhd$

Theorem 5.4. *Let S be a symmetric regular semi-Thue system over Σ, A be the mapping specifying the finite automata of S, and L be the REG^c logic corresponding to S. Let X and Γ be finite sets of formulas in NNF of the base language. Then Algorithm 1 is an* ExpTime *decision procedure for checking whether X is L-satisfiable w.r.t. the set Γ of global assumptions.*

Proof. It is easy to show that the algorithm has the invariant that a consistent marking of G cannot contain any node of $UnsatNodes$. The algorithm returns *false* only when the root v_0 belongs to $UnsatNodes$, i.e., only when G does not have any consistent marking. At Step 7b, G' is a marking of G that satisfies the local consistency property. If at that step $V = \emptyset$ then it satisfies also the global consistency property and is thus a consistent marking of G. That is, the algorithm returns *true* only when G has a consistent marking. Therefore, by Theorem 3.10, Algorithm 1 is a decision procedure for the considered problem. The complexity was established by Lemma 5.3. $\lhd$

As the problem of checking satisfiability in REG^c logics is ExpTime-complete [5], our algorithm is complexity-optimal for the considered problem.

5.2 Optimizations

Observe that Algorithm 1 first constructs an "and-or" graph and then checks whether the graph contains a consistent marking. To speed up the performance these two tasks can be done concurrently. For this we update the structures $UnsatNodes$, G', G_t of the algorithm "on-the-fly" during the construction of G. The main changes are:

- During the construction of the "and-or" graph G, each node of G has status *unexpanded*, *expanded*, *unsat* or *sat*. The initial status of a new node is *unexpanded*. When a node is expanded, we change its status to *expanded*. The status of a node changes to *unsat* (respectively *sat*) when there is an evidence that the contents of the node are unsatisfiable (respectively satisfiable) w.r.t. Γ. When a node becomes *unsat*, we insert it into the set $UnsatNodes$.
- When a node of G is expanded or G' is modified, we update G_t appropriately.
- When a new node is created, if its contents contain $\bot$ or a clashing pair φ, $\overline{\varphi}$ then we change the status of the node to *unsat*. This is the implicit application of the rule $(\bot_0)$ and a generalized form of the rule $(\bot)$. Thus, we can drop the explicit rules $(\bot_0)$ and $(\bot)$. When a non-empty set V of nodes of G becomes *unsat*, we call $updateUnsatNodes(G, UnsatNodes, V)$ to update the set $UnsatNodes$.
- When $UnsatNodes$ is modified, we update G' appropriately.
- Since G_t is not completed during the construction, when computing the set V of nodes of G' that cause G' not satisfying the global consistency property as in Step 7a of Algorithm 1 we treat a node (v, φ) of G_t also as an *end-node* if v has status *unexpanded* or *sat*.[14] We compute such a set V occasionally, accordingly to some criteria, and when

[14] If v has status *unexpanded* (respectively *sat*) then (v, φ) may (respectively must) be a productive node of G_t.

G_t has been completed. The computation is done by propagating "productiveness" backward through the graph G_t. The nodes of the resulting V become *unsat*.

During the construction of the "and-or" graph G, if a subgraph of G has been fully expanded in the sense that none of its nodes has status *unexpanded* or has a descendant node with status *unexpanded* then each node of the subgraph can be determined to be *unsat* or *sat* regardlessly of the rest of G. That is, if a node of the subgraph cannot be determined to be *unsat* by the operations described in the above list then we can set its status to *sat*. This technique was proposed in [24].

Recently, the first author has implemented a tableau prover called TGC (Tableau with Global Caching) [24] for checking consistency of a concept w.r.t. a TBox in the description logic $\mathcal{ALC}$. He has developed and implemented for TGC a special set of optimizations that co-operates very well with global caching and various search strategies on search spaces of the form "and-or" graph. Apart from search strategies and global caching for nodes of the constructed "and-or" graph, TGC also uses other optimizations like normalizing formulas, caching formulas using an efficient catalogue, simplification, semantic branching, propagation of *unsat* in a local scale using *unsat*-cores and subset-checking for parent nodes and brother nodes, as well as cutoffs. The test results of TGC on the sets T98-sat and T98-kb of DL'98 Systems Comparison are comparable with the test results of the best systems DLP-98 and FaCT-98 that took part in that comparison (see [24]). One can say that the mentioned test sets are not representative for practical applications, but the comparison at least shows that various optimization techniques can be applied together with global caching to significantly increase efficiency of tableau decision procedures for modal and description logics.

Most of the optimization techniques of TGC can be applied for our decision procedure for REG^c logics. There remains, however, the problem of cuts (not only of our calculus), as they can make the search space very large. Despite that the applicability of our rule (*cut*) is quite restricted, the rule is inflexible. It is possible that one can work out a more sophisticated condition for the applicability of the rule (*cut*). On the implementation level, we hope that depth-first search together with propagation of *unsat* for parent/brother nodes and cutoffs significantly reduces the negative side effects of cuts. If this is not the case, one can try to delay cuts in an appropriate way (preserving completeness).

6 Dealing with ABoxes

Using REG^c logics as description logics, possible worlds in a Kripke model, formulas and accessibility relations are regarded respectively as objects, concepts and roles. A set Γ of global assumptions is treated as a TBox. As for description logics, we introduce ABoxes and consider the problem of checking whether a given ABox is consistent with a given TBox, which is related to the instance checking problem.

We use the term *world variable* as an equivalent for the term "individual" used in description logic, and denote world variables by letters like a, b, c. We extend the notion of Kripke model so that the interpretation function h of a Kripke model M maps each world variable a to a world of M.

Definition 6.1.

- An *ABox* is a finite set of *assertions* of the form $a\!:\!\varphi$ or $\sigma(a,b)$, where φ is a formula in NNF of the base language. An ABox is *extensionally reduced* if it contains only assertions of the form $\sigma(a,b)$ or $a\!:\!p$, where $p \in \Phi_0$ is a proposition.
- A *TBox* is a finite set of formulas in NNF of the base language.
- A Kripke model $M = \langle W, R, h\rangle$ *satisfies* an ABox $\mathcal{A}$ if $M, h(a) \models \varphi$ for all $(a\!:\!\varphi) \in \mathcal{A}$ and $R_\sigma(h(a), h(b))$ holds for all $\sigma(a,b) \in \mathcal{A}$. An ABox $\mathcal{A}$ is *L-satisfiable w.r.t.* a TBox Γ iff there exists an L-model M that satisfies $\mathcal{A}$ and validates Γ. $\triangleleft$

We will refer to ABox assertions also as formulas. When necessary, we refer to formulas that are not ABox assertions as *formulas without world variables.*

6.1 A Tableau Calculus for the Satisfiability Checking Problem

In this subsection, we extend the calculus $\mathcal{C}L$ to calculus $\mathcal{C}L_{\mathrm{ABox}}$ for checking L-satisfiability of an ABox w.r.t. a TBox.

Definition 6.2. For a set X of formulas (possibly with world variables), the definition of $psf(X)$ remains unchanged, while $cl_L(X)$ is defined as follows:

$$\begin{aligned} cl_L(X) = psf(X) \cup \{&[A_\sigma, q]\varphi, \Box_\varrho[A_\sigma, q]\varphi, \langle A_\sigma, q\rangle\varphi, \Box_\varrho\langle A_\sigma, q\rangle\varphi, \langle\varrho\rangle\langle A_\sigma, q\rangle\varphi \mid \\ &\sigma, \varrho \in \Sigma, q \in Q_\sigma, \varphi \in psf(X), \text{ and} \\ &([\sigma]\varphi \in psf(X) \text{ or } [A_\sigma, q']\varphi \in X \text{ or } a\!:\![A_\sigma, q']\varphi \in X \text{ for some } q', a)\} \end{aligned} \quad \triangleleft$$

Notice that formulas of $psf(X)$ and $cl_L(X)$ do not contain world variables.

Definition 6.3. The *calculus* $\mathcal{C}L_{\mathrm{ABox}}$ w.r.t. a TBox Γ for the REG^c logic L extends the calculus $\mathcal{C}L$ with the following additional rules:

- a rule (ρ') obtained from each rule $(\rho) \in \{(\wedge), (\vee), (aut), ([\mathrm{A}]), (\langle \mathrm{A}\rangle), ([\mathrm{A}]_f), (\langle \mathrm{A}\rangle_f)\}$ by labeling the principal formula and the formulas obtained as its conclusions by prefix "a: ", for example:

$$(\vee')\ \frac{Y,\ a\!:\!\varphi \vee \psi}{Y,\ a\!:\!\varphi \vee \psi,\ a\!:\!\varphi \mid Y,\ a\!:\!\varphi \vee \psi,\ a\!:\!\psi}$$

- rules:

$$(\bot_0')\ \frac{Y,\ a\!:\!\bot}{\bot} \qquad\qquad (\bot')\ \frac{Y,\ a\!:\!p,\ a\!:\!\neg p}{\bot}$$

$$([\mathrm{A}]')\ \frac{Y,\ a\!:\!\Box_\sigma\varphi,\ \sigma(a,b)}{Y,\ a\!:\!\Box_\sigma\varphi,\ \sigma(a,b),\ b\!:\!\varphi}$$

$$(cut')\ \frac{Y}{Y, a\!:\![A_\sigma, q]\varphi \mid Y, a\!:\!\Box_\varrho\langle A_\sigma, q'\rangle\overline{\varphi}}\ \text{ if } (*)$$

(*) : if Y contains some formula $a\!:\!\langle\varrho\rangle\psi$ or $\varrho(a,b)$, and $[A_\sigma, q']\varphi$ belongs to $cl_L(Y \cup \Gamma)$, and $(q', \overline{\varrho}, q) \in \delta_\sigma$

$$(trans')\ \frac{Y}{\&\{\ (\{\varphi\} \cup \{\psi \text{ s.t. } (a\!:\!\Box_\sigma\psi) \in Y\} \cup \Gamma) \text{ s.t. } (a\!:\!\langle\sigma\rangle\varphi) \in Y\ \}}$$

The rule $(trans')$ is an "and"-rule and a transitional rule. The other additional rules of $\mathcal{C}L_{\text{ABox}}$ are "or"-rules and static rules. A rule among the additional static rules of $\mathcal{C}L_{\text{ABox}}$ except $(\bot_0')$, $(\bot')$, $(\langle\text{A}\rangle')$, $(\langle\text{A}\rangle_f')$ is applicable only when all of its possible conclusions are proper supersets of its premise.[15] $\lhd$

The additional rules of $\mathcal{C}L_{\text{ABox}}$ work on sets of ABox assertions, except that the conclusions of $(trans')$ are sets of formulas without world variables. That is, in those rules, Y denotes a set of ABox assertions.

Similarly as for $\mathcal{C}L$, we assume the following preferences for the rules of $\mathcal{C}L_{\text{ABox}}$: the rules $(\bot_0)$, $(\bot)$, $(\bot_0')$, $(\bot')$ have the highest priority; unary static rules have a higher priority than non-unary static rules; the rules (cut) and (cut') have the lowest priority among static rules; all the static rules have a higher priority than the transitional rules.

Definition 6.4. Let $\mathcal{A}$ be an ABox and Γ be a TBox. An *"and-or" graph* for $(\mathcal{A}, \Gamma)$ w.r.t. $\mathcal{C}L_{\text{ABox}}$ is constructed according to the following principles:

- The graph contains nodes of two kinds: *complex nodes* and *simple nodes*.
- The contents of a complex node consist of ABox assertions, while the contents of a simple node consist of formulas without world variables.
- The graph never contains edges from a simple node to a complex node.
- The root of the graph is a complex node with contents $\mathcal{A} \cup \{(a\!:\!\varphi) \mid \varphi \in \Gamma$ and a is a world variable occurring in $\mathcal{A}\}$.
- Complex nodes are expanded using the additional rules of $\mathcal{C}L_{\text{ABox}}$ (the primed rules), while simple nodes are expanded using the rules of $\mathcal{C}L$.
- The "and-or" graph is expanded in the same way as described in the previous section for checking L-satisfiability of a set X of formulas w.r.t. Γ. The only exception is that, instead of $(trans)$ we may use $(trans')$, depending on whether the expanded node is a complex node or a simple node. $\lhd$

Example 6.5. In Figure 4 we present an "and-or" graph for $(\{a\!:\![\sigma]p,\ b\!:\!\neg p,\ \overline{\sigma}(b,a)\}, \emptyset)$ w.r.t. $\mathcal{C}L_{\text{ABox}}$, where L is the REG^c logic corresponding to the empty semi-Thue system and $A_\sigma = \langle\{\sigma,\overline{\sigma}\},\ \{0,1\},\ \{0\},\ \{(0,\sigma,1)\},\ \{1\}\rangle$. The nodes are numbered when created and are expanded using DFS. In each node, we display the formulas of the contents of the node and the name of the rule expanding the node. The edge $((9),(10))$ is labeled by $a\!:\!\langle\sigma\rangle\langle A_\sigma, 1\rangle\neg p$. The example demonstrates the usage of (cut'). $\lhd$

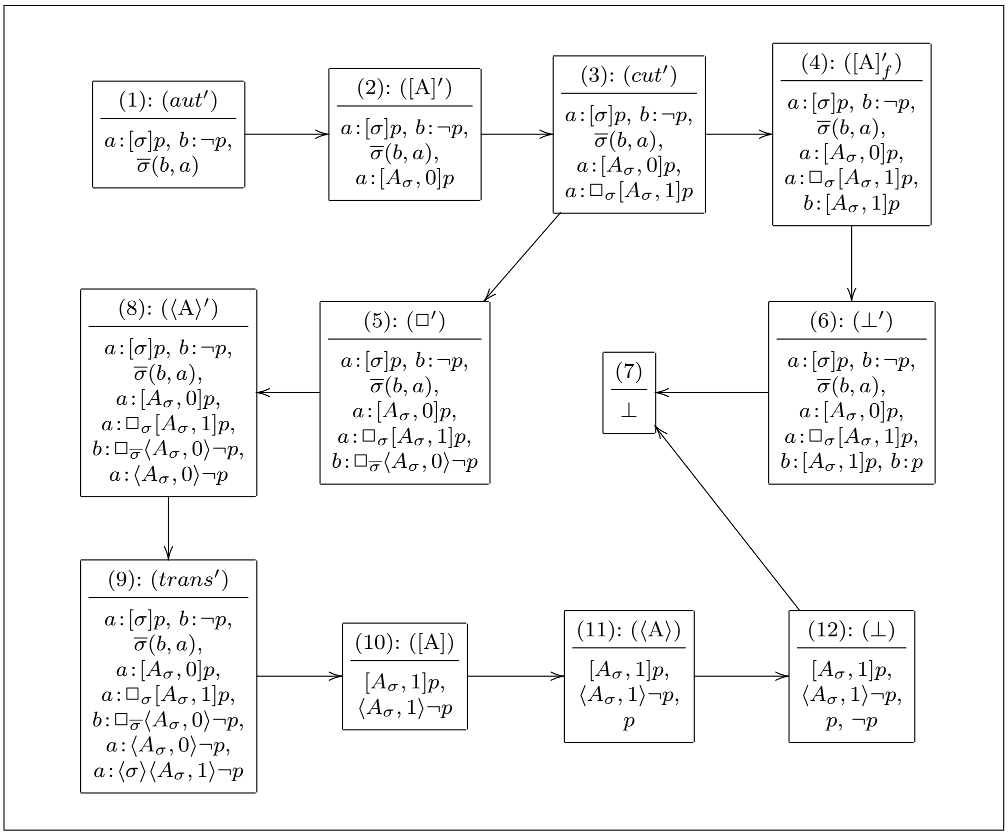
The notion of *marking* remains unchanged. The notions of *trace* and $\Diamond$*-realization* also remain unchanged but are defined only for formulas without world variables (of the form $\langle A_\sigma, q\rangle\varphi$) of the simple nodes.

Definition 6.6. A marking G' of an "and-or" graph G is *consistent* if:

- *local consistency:* G' does not contain any node with contents $\{\bot\}$,
- *global consistency:* for every *simple* node v of G', every formula of the form $\langle A_\sigma, q\rangle\varphi$ of the contents of v has a $\Diamond$-realization (starting from v) in G'. $\lhd$

Notice that the only change for the above definition is that global consistency refers to "simple" nodes only.

[15] Those rules are thus monotonic.



(1): (aut')
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a)$
(2): $([A]')$
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a),\ a\!:[A_\sigma,0]p$
(3): (cut')
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a),\ a\!:[A_\sigma,0]p,\ a\!:\Box_\sigma[A_\sigma,1]p$
(4): $([A]'_f)$
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a),\ a\!:[A_\sigma,0]p,\ a\!:\Box_\sigma[A_\sigma,1]p,\ b\!:[A_\sigma,1]p$
(5): $(\Box')$
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a),\ a\!:[A_\sigma,0]p,\ a\!:\Box_\sigma[A_\sigma,1]p,\ b\!:\Box_{\overline{\sigma}}\langle A_\sigma,0\rangle\neg p$
(6): $(\bot')$
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a),\ a\!:[A_\sigma,0]p,\ a\!:\Box_\sigma[A_\sigma,1]p,\ b\!:[A_\sigma,1]p,\ b\!:p$
(7)
$\bot$
(8): $(\langle A\rangle')$
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a),\ a\!:[A_\sigma,0]p,\ a\!:\Box_\sigma[A_\sigma,1]p,\ b\!:\Box_{\overline{\sigma}}\langle A_\sigma,0\rangle\neg p,\ a\!:\langle A_\sigma,0\rangle\neg p$
(9): $(trans')$
$a\!:[\sigma]p,\ b\!:\neg p,\ \overline{\sigma}(b,a),\ a\!:[A_\sigma,0]p,\ a\!:\Box_\sigma[A_\sigma,1]p,\ b\!:\Box_{\overline{\sigma}}\langle A_\sigma,0\rangle\neg p,\ a\!:\langle A_\sigma,0\rangle\neg p,\ a\!:\langle\sigma\rangle\langle A_\sigma,1\rangle\neg p$
(10): $([A])$
$[A_\sigma,1]p,\ \langle A_\sigma,1\rangle\neg p$
(11): $(\langle A\rangle)$
$[A_\sigma,1]p,\ \langle A_\sigma,1\rangle\neg p,\ p$
(12): $(\bot)$
$[A_\sigma,1]p,\ \langle A_\sigma,1\rangle\neg p,\ p,\ \neg p$

Fig. 4. An "and-or" graph considered in Example 6.5.

Example 6.7. In description logic, the symbols $\sqcap$, $\sqcup$, $\sqsubseteq$ are used instead of $\wedge$, $\vee$, $\rightarrow$. Furthermore, formulas $[\sigma]\varphi$ and $\langle\sigma\rangle\varphi$ are written as $\forall\sigma.\varphi$ and $\exists\sigma.\varphi$, respectively. This example is about web pages. It is formulated in the description logic $\mathcal{ALC}$, a notational variant of the REG^c logic that corresponds to the empty semi-Thue system. Let

$$\Gamma = \{perfect \sqsubseteq interesting \sqcap \forall link.perfect\}$$
$$\mathcal{A} = \{a\!:\!perfect,\ link(a,b)\}.$$

It can be shown that b is an instance of the concept $\forall link.interesting$ w.r.t. the knowledge base $(\Gamma,\mathcal{A})$ in $\mathcal{ALC}$. That is, for every Kripke model $M = \langle W,R,h\rangle$ that satisfies $\mathcal{A}$ and validates Γ, we have that $M,h(b) \models \forall link.interesting$. To prove this one can show that $\mathcal{A}\cup\{b\!:\!\exists link.\neg interesting\}$ is $\mathcal{ALC}$-unsatisfiable w.r.t. Γ.

As abbreviations, let $p = perfect$, $q = interesting$, $\sigma = link$, and let φ denotes the only formula of Γ. In Figure 5 we present an "and-or" graph for $(\{a\!:\!p,\ \sigma(a,b),\ b\!:\!\langle\sigma\rangle\neg q\},\ \{\varphi\})$, where $\varphi = \neg p \vee (q \wedge [\sigma]p)$ and $A_\sigma = \langle\{\sigma,\overline{\sigma}\},\ \{0,1\},\ \{0\},\ \{(0,\sigma,1)\},\ \{1\}\rangle$. The nodes are numbered when created and are expanded using DFS. In each node, we display the formulas of the contents of the node and the name of the rule expanding the node. The edge $((14),(15))$ is labeled by $b\!:\!\langle\sigma\rangle\neg q$.

As the graph has no consistent marking, by Theorem 6.8 given below, $\{a\!:\!p,\ \sigma(a,b),\ b\!:\!\langle\sigma\rangle\neg q\}$ is $\mathcal{ALC}$-unsatisfiable w.r.t. $\{\varphi\}$. $\lhd$

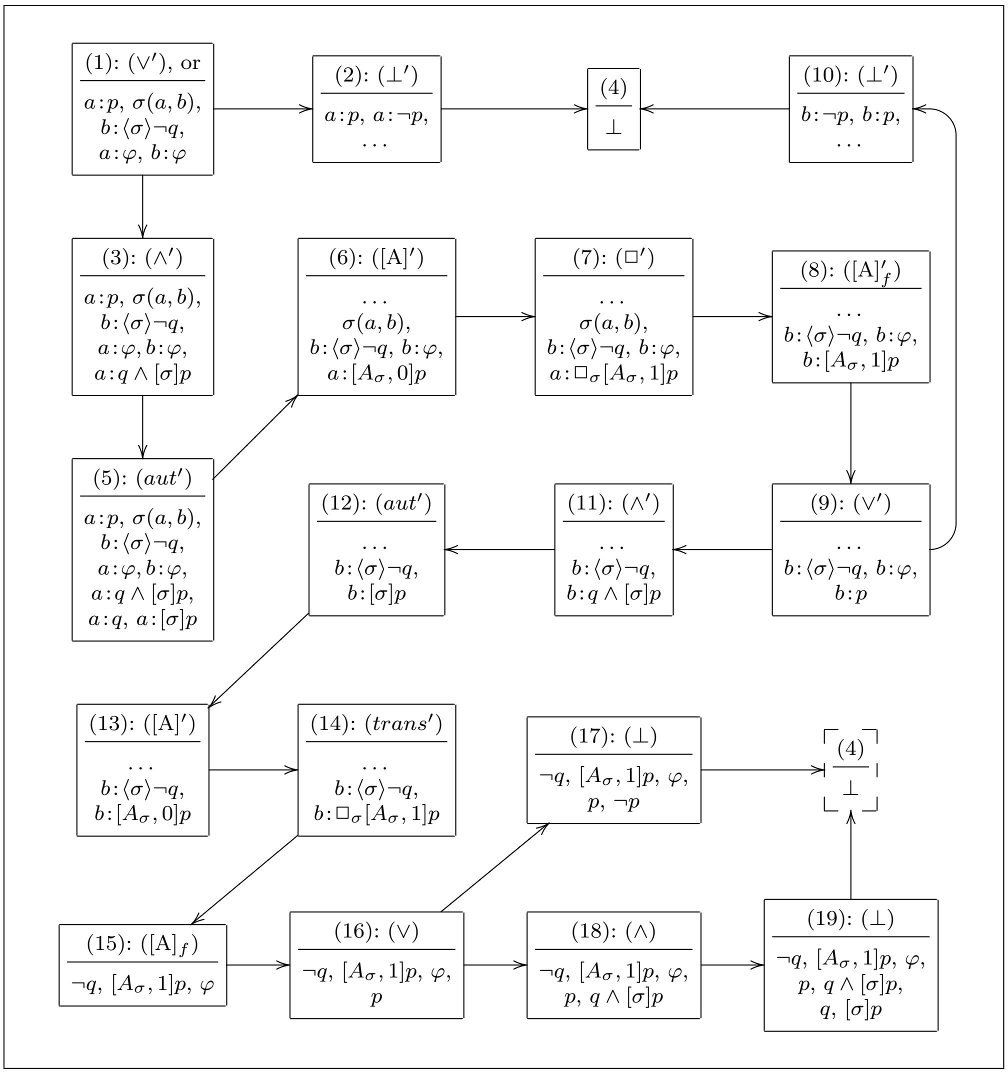

(1): $(\vee')$, or
$a{:}p$, $\sigma(a,b)$, $b{:}\langle\sigma\rangle\neg q$, $a{:}\varphi$, $b{:}\varphi$
(2): $(\bot')$
$a{:}p$, $a{:}\neg p$, ...
(4)
$\bot$
(10): $(\bot')$
$b{:}\neg p$, $b{:}p$, ...
(3): $(\wedge')$
$a{:}p$, $\sigma(a,b)$, $b{:}\langle\sigma\rangle\neg q$, $a{:}\varphi, b{:}\varphi$, $a{:}q \wedge [\sigma]p$
(6): $([\mathrm{A}]')$
..., $\sigma(a,b)$, $b{:}\langle\sigma\rangle\neg q$, $b{:}\varphi$, $a{:}[A_\sigma,0]p$
(7): $(\Box')$
..., $\sigma(a,b)$, $b{:}\langle\sigma\rangle\neg q$, $b{:}\varphi$, $a{:}\Box_\sigma[A_\sigma,1]p$
(8): $([\mathrm{A}]'_f)$
..., $b{:}\langle\sigma\rangle\neg q$, $b{:}\varphi$, $b{:}[A_\sigma,1]p$
(5): (aut')
$a{:}p$, $\sigma(a,b)$, $b{:}\langle\sigma\rangle\neg q$, $a{:}\varphi, b{:}\varphi$, $a{:}q \wedge [\sigma]p$, $a{:}q$, $a{:}[\sigma]p$
(12): (aut')
..., $b{:}\langle\sigma\rangle\neg q$, $b{:}[\sigma]p$
(11): $(\wedge')$
..., $b{:}\langle\sigma\rangle\neg q$, $b{:}q \wedge [\sigma]p$
(9): $(\vee')$
..., $b{:}\langle\sigma\rangle\neg q$, $b{:}\varphi$, $b{:}p$
(13): $([\mathrm{A}]')$
..., $b{:}\langle\sigma\rangle\neg q$, $b{:}[A_\sigma,0]p$
(14): $(trans')$
..., $b{:}\langle\sigma\rangle\neg q$, $b{:}\Box_\sigma[A_\sigma,1]p$
(17): $(\bot)$
$\neg q$, $[A_\sigma,1]p$, φ, p, $\neg p$
(4)
$\bot$
(15): $([\mathrm{A}]_f)$
$\neg q$, $[A_\sigma,1]p$, φ
(16): $(\vee)$
$\neg q$, $[A_\sigma,1]p$, φ, p
(18): $(\wedge)$
$\neg q$, $[A_\sigma,1]p$, φ, p, $q \wedge [\sigma]p$
(19): $(\bot)$
$\neg q$, $[A_\sigma,1]p$, φ, p, $q \wedge [\sigma]p$, q, $[\sigma]p$

Fig. 5. An "and-or" graph considered in Example 6.7

Theorem 6.8 (Soundness and Completeness of $\mathcal{C}L_{\mathrm{ABox}}$). *Let S be a symmetric regular semi-Thue system over Σ, A be the mapping specifying the finite automata of S, and L be the REG^c logic corresponding to S. Let $\mathcal{A}$ be an ABox, Γ a TBox, and G an "and-or" graph for $(\mathcal{A}, \Gamma)$ w.r.t. $\mathcal{C}L_{\mathrm{ABox}}$. Then $\mathcal{A}$ is L-satisfiable w.r.t. Γ iff G has a consistent marking.* $\lhd$

The "only if" direction means soundness of $\mathcal{C}L_{\mathrm{ABox}}$, while the "if" direction means completeness of $\mathcal{C}L_{\mathrm{ABox}}$. This theorem follows from Lemmas 6.9 and 6.10 given below.

Lemma 6.9. *Let S, A, L, $\mathcal{A}$, Γ, G be as described in Theorem 6.8. Suppose that $\mathcal{A}$ is L-satisfiable w.r.t. Γ. Then G has a consistent marking.*

Proof. We construct a consistent marking G' of G as follows. At the beginning, G' contains only the root of G. Then, for every node v of G' and for every successor w of v in G, if either w is a complex node and the contents of w are an ABox L-satisfiable w.r.t. Γ, or w is a simple node and the contents of w are a set of formulas that is L-satisfiable w.r.t. Γ, then add the node w and the edge (v, w) to G'. The proof of that G' is a consistent marking of G is similar to the proof of Lemma 4.1. $\lhd$

Lemma 6.10. *Let S, A, L, $\mathcal{A}$, Γ, G be as described in Theorem 6.8. Suppose that G has a consistent marking G'. Then $\mathcal{A}$ is L-satisfiable w.r.t. Γ.*

Proof. We construct a model graph $M = \langle W, R, H\rangle$ as follows:

1. Let v_0 be the root of G' and $v_0, \dots, v_k$ be a saturation path of v_0 w.r.t. G'. Let W_0 to the set of all world variables occurring in $\mathcal{A}$ and set $W = W_0$. For each $a \in W_0$, set $H(a)$ to the set of all φ such that $a\!:\!\varphi$ belongs to the contents of some v_i $(0 \leq i \leq k)$, and mark a as *unresolved*. (Each node of M will be marked either as unresolved or as resolved.) For each $\sigma \in \Sigma$, set $R_\sigma = \{(a, b) \mid \sigma(a, b) \in \mathcal{A}\}$.
2. While W contains unresolved nodes, take one unresolved node w_0 and do:
   (a) For every formula $\langle\varrho\rangle\varphi \in H(w_0)$ do:
      i. Let $\varphi_0 = \langle\varrho\rangle\varphi$ and $\varphi_1 = \varphi$.
      ii. A. If $w_0 \in W_0$ then:
         – Let $u_0 = v_k$.
         – Let u_1 be the node of G' such that the edge (u_0, u_1) is labeled by $(w_0\!:\!\varphi_0)$. (Recall that w_0 is a world variable and note that φ_1 belongs to the contents of u_1.)
      B. Else:
         – Let $u_0 = f(w_0)$. (f is a constructed mapping that maps each node of M not belonging to W_0 to an "and"-node of G'. As a maintained property of f, φ_0 belongs to the contents of u_0.)
         – Let u_1 be the node of G' such that the edge (u_0, u_1) is labeled by φ_0. (Note that φ_1 belongs to the contents of u_1.)
      iii. If φ is of the form $\langle A_\sigma, q\rangle\psi$ then:
      A. Let $(u_1, \varphi_1), \dots, (u_l, \varphi_l)$ be a $\Diamond$-realization in G' for φ_1 at u_1.
      B. Let $u_l, \dots, u_m$ be a saturation path of u_l w.r.t. G'.
      iv. Else let $u_1, \dots, u_m$ be a saturation path of u_1 w.r.t. G'.
      v. Let $j_0 = 0 < j_1 < \ldots < j_{n-1} < j_n = m$ be all the indices such that, for $0 \leq j \leq m$, u_j is an "and"-node of G iff $j \in \{j_0, \dots, j_n\}$. Let $\varrho_0 = \varrho$. For $1 \leq s \leq n - 1$, let $\langle\varrho_s\rangle\varphi_{j_s+1}$ be the label of the edge (u_{j_s}, u_{j_s+1}) of G'.
      vi. For $1 \leq s \leq n$ do:
      A. Let Z_s be the sum of the contents of the nodes $u_{j_{s-1}+1}, \dots, u_{j_s}$.
      B. If there does not exist $w_s \in W$ such that $H(w_s) = Z_s$ then: add a new node w_s to W, set $H(w_s) = Z_s$, mark w_s as unresolved, and set $f(w_s) = u_{j_s}$.
      C. Add the pair (w_{s-1}, w_s) to $R_{\varrho_{s-1}}$.
   (b) Mark w_0 as resolved.

Note that the above construction differs from the construction given in the proof of Lemma 4.11 mainly by Steps 1 and 2(a)iiA.

The above construction terminates and results in a finite model graph since for every $w, w' \in W \setminus W_0$, $w \neq w'$ implies $H(w) \neq H(w')$, and for every $w \in W$, $H(w)$ is a subset of $cl_L(\mathcal{A} \cup \Gamma)$.

Similarly as for the construction given in the proof of Lemma 4.11, it can be seen that M is a consistent and $\mathcal{C}L$-saturated model graph.

It can be seen that: if $(a\!:\!\varphi) \in \mathcal{A}$ then $\varphi \in H(a)$; if $\sigma(a,b) \in \mathcal{A}$ then $(a,b) \in R_\sigma$; and $\Gamma \subseteq H(w)$ for all $w \in W$. Hence, by Lemma 4.9, the Kripke model corresponding to M validates Γ and satisfies $\mathcal{A}$. Thus $\mathcal{A}$ is L-satisfiable w.r.t. Γ. $\lhd$

6.2 Checking *L*-Satisfiability of an ABox w.r.t. a TBox

Let $\mathcal{A}$ be an ABox, Γ a TBox, G an "and-or" graph for $(\mathcal{A},\Gamma)$ w.r.t. $\mathcal{C}L_{\mathrm{ABox}}$, and G' a marking of G. The graph G_t of traces of G' in G is defined as in Section 5.1 with the exception that nodes of G_t are pairs (v,φ), where v is a simple node of G and φ is a formula of the contents of v.

By **Algorithm 1′** we understand the algorithm obtained from Algorithm 1 by changing X to $\mathcal{A}$. Algorithm 1′ receives an ABox $\mathcal{A}$, a TBox Γ and the mapping A specifying the finite automata of a REG^c logic L as input and checks whether $\mathcal{A}$ is L-satisfiable w.r.t. Γ.

Here is a counterpart of Lemma 5.2:

Lemma 6.11. *Let*

- *S be a symmetric regular semi-Thue system over Σ,*
- *A be the mapping specifying the finite automata of S,*
- *L be the REG^c logic corresponding to S,*
- *$\mathcal{A}$ be an ABox and Γ be a TBox,*
- *G be an "and-or" graph for $(\mathcal{A},\Gamma)$ w.r.t. $\mathcal{C}L_{\mathrm{ABox}}$,*
- *$l = |\Sigma|$,*
- *m be the sum of the sizes of the automata A_σ for $\sigma \in \Sigma$,*
- *n be the size of $\mathcal{A} \cup \Gamma$.*

Then

- *G has $2^{O(l \times m \times n^2)}$ nodes,*
- *the contents of each simple node of G have $O(l \times m \times n)$ formulas and are of size $O(l \times m \times n^2)$,*
- *the contents of each complex node of G have $O(l \times m \times n^2)$ formulas and are of size $O(l \times m \times n^3)$.*

Proof. Let V be the set of all world variables occurring in $\mathcal{A}$ and let

$$X = \Gamma \cup \{\varphi \mid (a\!:\!\varphi) \in \mathcal{A} \text{ for some } a \in V\}.$$

The set $cl_L(X)$ has $O(l \times m \times n)$ formulas. For the second assertion, just notice that the contents of each simple node of G are a subset of $cl_L(X)$. Since the contents of simple nodes are unique, G has $2^{O(l \times m \times n)}$ simple nodes. For each complex node v of G and for each $a \in V$, the set $\{\varphi \mid (a\!:\!\varphi)$ belongs to the contents of $v\}$ is also a subset of $cl_L(X)$. Hence the contents of each complex node of G contain $O(l \times m \times n^2)$ formulas and is of size $O(l \times m \times n^3)$. Observe that each path of complex nodes in G has length of rank $O(l \times m \times n^2)$. Hence G has $2^{O(l \times m \times n^2)}$ complex nodes. $\lhd$

Theorem 6.12. *Algorithm 1′ is an* ExpTime *decision procedure for checking whether a given ABox $\mathcal{A}$ is satisfiable w.r.t. a given TBox Γ in a REG^c logic.*

Proof. Use the proofs of Lemma 5.3 and Theorem 5.4 with appropriate changes. $\triangleleft$

Corollary 6.13. *The problem of checking satisfiability of an ABox w.r.t. a TBox in a REG^c logic is* ExpTime*-complete.* $\triangleleft$

Algorithm 1′ uses global caching for both complex nodes and simple nodes. The rest of this subsection deals with the following questions:

- what happens if we use global caching only for simple nodes and backtracking on branchings at complex "or"-nodes?
- is the complexity still ExpTime?

Lemma 6.14. *Let $\mathcal{A}$ be an ABox, Γ a TBox, and G an "and-or" graph for $(\mathcal{A}, \Gamma)$ w.r.t. $\mathcal{CL}_{\text{ABox}}$. Then G has a consistent marking iff there exists a complex "and"-node v of G such that the subgraph generated by v of G (which uses v as the root) has a consistent marking.*

Proof. Just notice that the root of G is a complex node and every father node of a complex node must be a complex "or"-node. $\triangleleft$

By **Algorithm 1″** we understand the algorithm that checks whether a given ABox $\mathcal{A}$ is L-satisfiable w.r.t. a given TBox Γ as follows. The algorithm simulates the tasks of constructing an "and-or" graph for $(\mathcal{A}, \Gamma)$ w.r.t. $\mathcal{CL}_{\text{ABox}}$ and checking whether the graph has a consistent marking but does it as follows:

1. nondeterministically expand a path from the root until reaching a complex "and"-node v,
2. construct the full subgraph rooted at v,
3. check whether the subgraph has a consistent marking (as done in the steps 2–7 of Algorithm 1), and return *true* if it does,
4. if none of the possible executions returns *true* then return *false*.

In practice, the first step of the above algorithm is executed by backtracking on the branchings of the applications of "or"-rules. The algorithm does not keep all complex nodes but only the ones on the current path of complex nodes. On the other hand, simple nodes can be globally cached. That is, simple nodes can be left through backtracking for use in the next possible executions.

Theorem 6.15. *Using backtracking to deal with nondeterminism, Algorithm 1″ is an* ExpTime *decision procedure for checking whether a given ABox $\mathcal{A}$ is satisfiable w.r.t. a given TBox Γ in a REG^c logic.*

Proof. By Theorem 6.8 and Lemma 6.14, Algorithm 1″ is a decision procedure for the considered problem. It remains to show that the algorithm runs in exponential time. Let $l = |\Sigma|$, m be the sum of the sizes of the finite automata of the considered REG^c logic, and n be the size of $\mathcal{A} \cup \Gamma$.

As stated in the proof of Lemma 6.11, each path of complex nodes constructed by Step 1 of Algorithm $1''$ has length of rank $O(l \times m \times n^2)$. Analogously to the proofs of Lemmas 5.3 and 6.11, it can be shown that Steps 2 and 3 of Algorithm $1''$ are executed in $2^{O(l \times m \times n)}$ steps. Hence the complexity of Algorithm $1''$ is of rank $2^{O(l \times m \times n^2)} \times 2^{O(l \times m \times n)}$, which is $2^{O(l \times m \times n^2)}$. $\lhd$

6.3 On the Instance Checking Problem

Consider the use of a *REG*^{c} logic L as a description logic. A pair $(\mathcal{A}, \mathit{\Gamma})$ of an ABox $\mathcal{A}$ and a TBox $\mathit{\Gamma}$ is treated as a knowledge base. An L-model that satisfies $\mathcal{A}$ and validates $\mathit{\Gamma}$ is called an L-model of $(\mathcal{A}, \mathit{\Gamma})$. Given a formula φ without world variables, which is treated as a "concept", and a world variable a, which is treated as an "individual", the problem of checking whether $M, h(a) \models \varphi$ in every L-model $M = \langle W, R, h \rangle$ of $(\mathcal{A}, \mathit{\Gamma})$ is called the instance checking problem in L. Denote the condition to check by $(\mathcal{A}, \mathit{\Gamma}) \models_L \varphi(a)$.

Observe that $(\mathcal{A}, \mathit{\Gamma}) \models_L \varphi(a)$ iff the ABox $\mathcal{A} \cup \{a : \overline{\varphi}\}$ is L-unsatisfiable w.r.t. $\mathit{\Gamma}$. So, the instance checking problem is reduced to the problem of checking L-unsatisfiability of an ABox w.r.t. a TBox. What we are interested in is the *data complexity* of the instance checking problem, which is measured in the size of $\mathcal{A}$ when assuming that $\mathcal{A}$ is extensionally reduced and L, $\mathit{\Gamma}$, φ, a are fixed. Here, L, $\mathit{\Gamma}$, φ and a form a fixed query, while $\mathcal{A}$ varies as input data.

Theorem 6.16. *The data complexity of the instance checking problem in REG^c is coNP-complete.*

Proof. Let $\mathcal{A}$ be an extensionally reduced ABox, $\mathit{\Gamma}$ a TBox, φ a formula in NNF of the base language, and a a world variable. Consider the problem of checking whether $(\mathcal{A}, \mathit{\Gamma}) \models_L \varphi(a)$.

Let p be a fresh proposition (not occurring in $\mathcal{A}$, $\mathit{\Gamma}$, φ) and let

$$\begin{aligned} \mathit{\Gamma}' &= \mathit{\Gamma} \cup \{\neg p \lor \varphi, p \lor \overline{\varphi}\} \\ \mathcal{A}' &= \mathcal{A} \cup \{a : \neg p\}. \end{aligned}$$

Observe that $\mathit{\Gamma}'$ extends $\mathit{\Gamma}$ with the formulas stating that p is equivalent to φ, and that $(\mathcal{A}, \mathit{\Gamma}) \models_L \varphi(a)$ iff the ABox $\mathcal{A}'$ is L-unsatisfiable w.r.t. the TBox $\mathit{\Gamma}'$.

Let n be the size of $\mathcal{A}$. The size of $\mathcal{A}' \cup \mathit{\Gamma}'$ is thus of rank $O(n)$.

Consider an execution of Algorithm $1''$ for the pair $\mathcal{A}'$ and $\mathit{\Gamma}'$. As stated in the proof of Lemma 6.11, each path of complex nodes constructed by Step 1 of Algorithm $1''$ has length of rank $O(n^2)$ (as l and m mentioned there are constants). Each complex node has contents of size $O(n^3)$. Since A' is extensionally reduced, the contents of each created simple node depend only on $\mathit{\Gamma}'$. Since $\mathit{\Gamma}'$ is fixed, Steps 2 and 3 of Algorithm $1''$ are executed in time of rank $O(n^2)$. Hence a nondeterministic execution of Step 1 of Algorithm $1''$ runs in time $O(n^2) \times O(n^3) + O(n^2)$. It follows that the execution of Algorithm $1''$ for $\mathcal{A}'$ and $\mathit{\Gamma}'$ runs nondeterministically in polynomial time the size of $\mathcal{A}$. Therefore the instance checking problem $(\mathcal{A}, \mathit{\Gamma}) \models_L \varphi(a)$ is in coNP.

The coNP-hardness follows from the fact that the instance checking problem in the description logic $\mathcal{ALC}$ is coNP-hard [28]. $\lhd$

7 Conclusions

In this paper we have provided sound and complete tableau calculi for the general satisfiability problem of REG^c logics and the problem of checking consistency of an ABox w.r.t. a TBox in a REG^c logic. The results are novel, since up to now no tableau calculi have been fully developed for REG^c logics. Using the calculi we have developed the first complexity-optimal tableau decision procedures not based on transformation for the mentioned problems, to which a number of useful optimization techniques can be applied. We have also proved the new result that the data complexity of the instance checking problem in REG^c logics is coNP-complete.

References

1. M. Baldoni, L. Giordano, and A. Martelli. A tableau for multimodal logics and some (un)decidability results. In *Proceedings of TABLEAUX'1998, LNCS 1397:44-59*, 1998.
2. G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 117-137:87–138, 2000.
3. S. Demri. The complexity of regularity in grammar logics and related modal logics. *Journal of Logic and Computation*, 11(6):933–960, 2001.
4. S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. arXiv:cs.LO/0306117, 2004.
5. S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language and Information*, 14(3):289–329, 2005.
6. F. Donini and F. Massacci. ExpTime tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124:87–138, 2000.
7. L. Fariñas del Cerro and M. Penttonen. Grammar logics. *Logique et Analyse*, 121-122:123–134, 1988.
8. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* volume 169 of Synthese Library. D. Reidel, Dordrecht, Holland, 1983.
9. G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms.* PhD thesis, Universita' di Roma "La Sapienza", 1995.
10. R. Goré. Tableau methods for modal and temporal logics. In D'Agostino et al, editor, *Handbook of Tableau Methods*, pages 297–396. Kluwer, 1999.
11. R. Goré and L.A. Nguyen. A tableau system with automaton-labelled formulae for regular grammar logics. In B. Beckert, editor, *Proceedings of TABLEAUX 2005, LNAI 3702*, pages 138–152. Springer-Verlag, 2005.
12. R. Goré and L.A. Nguyen. ExpTime tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In N. Olivetti, editor, *Proc. of TABLEAUX 2007, LNAI 4548*, pages 133–148. Springer-Verlag, 2007.
13. R. Goré and L.A. Nguyen. Analytic cut-free tableaux for regular modal logics of agent beliefs. In F. Sadri and K. Satoh, editors, *Proceedings of CLIMA VIII, LNAI 5056*, pages 268–287. Springer-Verlag, 2008.
14. R. Goré and L.A. Nguyen. Sound global caching for abstract modal tableaux. In H.-D. Burkhard et al, editor, *Proceedings of CS&P'2008*, pages 157–167, 2008.
15. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* MIT Press, 2000.
16. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *Proceedings of KR'2006*, pages 57–67. AAAI Press, 2006.
17. I. Horrocks and P.F. Patel-Schneider. Optimizing description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
18. I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 160(1-2):79–104, 2004.
19. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In L.P. Kaelbling and A. Saffiotti, editors, *Proceedings of IJCAI-05*, pages 466–471. Professional Book Center, 2005.
20. A. Mateescu and A. Salomaa. Formal languages: an introduction and a synopsis. In *Handbook of Formal Languages - Volume 1*, pages 1–40. Springer, 1997.

21. L.A. Nguyen. Analytic tableau systems and interpolation for the modal logics KB, KDB, K5, KD5. *Studia Logica*, 69(1):41–57, 2001.
22. L.A. Nguyen. On the deterministic Horn fragment of test-free PDL. In I. Hodkinson and Y. Venema, editors, *Advances in Modal Logic - Volume 6*, pages 373–392. King's College Publications, 2006.
23. L.A. Nguyen. Weakening Horn knowledge bases in regular description logics to have PTIME data complexity. In Ghilardi et al, editor, *Proceedings of ADDCT'07* (see also the extension at `http://www.mimuw.edu.pl/~nguyen/papers.html`), pages 32–47, 2007.
24. L.A. Nguyen. An efficient tableau prover using global caching for the description logic ALC. *Fundamenta Informaticae*, 93(1-3):273–288, 2009.
25. L.A. Nguyen and A. Szałas. A tableau calculus for regular grammar logics with converse. In R.A. Schmidt, editor, *Proceedings of CADE-22, LNAI 5663*, pages 421–436. Springer-Verlag, 2009.
26. V.R. Pratt. A near-optimal method for reasoning about action. *J. Comput. Syst. Sci.*, 20(2):231–254, 1980.
27. W. Rautenberg. Modal tableau calculi and interpolation. *JPL*, 12:403–423, 1983.
28. A. Schaerf. Reasoning with individuals in concept languages. *Data Knowl. Eng.*, 13(2):141–176, 1994.