Grammar Logics in Nested Sequent Calculus: Proof Theory and Decision Procedures

Alwen Tiu

Research School of Computer Science, The Australian National University

Egor Ianovski

Department of Computer Science, University of Auckland

Rajeev Goré

Research School of Computer Science, The Australian National University

Abstract

A grammar logic refers to an extension of the multi-modal logic K in which the modal axioms are generated from a formal grammar. We consider a proof theory, in nested sequent calculus, of grammar logics with converse, i.e., every modal operator $[a]$ comes with a converse $[\bar{a}]$. Extending previous works on nested sequent systems for tense logics, we show all grammar logics (with or without converse) can be formalised in nested sequent calculi, where the axioms are internalised in the calculi as structural rules. Syntactic cut-elimination for these calculi is proved using a procedure similar to that for display logics. If the grammar is context-free, then one can get rid of all structural rules, in favor of deep inference and additional propagation rules. We give a novel semi-decision procedure for context-free grammar logics, using nested sequent calculus with deep inference, and show that, in the case where the given context-free grammar is regular, this procedure terminates. Unlike all other existing decision procedures for regular grammar logics in the literature, our procedure does not assume that a finite state automaton encoding the axioms is given.

Keywords: Nested sequent calculus, display calculus, modal logics, deep inference.

1 Introduction

A grammar logic refers to an extension of the multi-modal logic K in which the modal axioms are generated from a formal grammar. Thus given a set Σ of indices, and a grammar production rule as shown below left, where each a_i and b_j are in Σ, we extend K with the multi-modal axiom shown below right:

$$a_1 a_2 \cdots a_l \to b_1 b_2 \cdots b_r \qquad\qquad [a_1][a_2]\cdots[a_l]A \supset [b_1][b_2]\cdots[b_r]A$$

The logic is a context-free grammar logic if $l = 1$ and furthermore, is a right linear grammar logic if the production rules also define a right linear grammar.

The logic is a regular grammar logic if the set of words generated from each $a \in \Sigma$ using the grammar production rules is a regular language. A right linear grammar logic is also a regular grammar logic since a right linear grammar can be converted to a finite automaton in polynomial time. Adding "converse" gives us alphabet symbols like $\bar{a}$ which correspond to the converse modality $[\bar{a}]$ and lead to multi-modal extensions of tense logic Kt where each modality $[a]$ and its converse $[\bar{a}]$ obey the interaction axioms $A \supset [a]\langle\bar{a}\rangle A$ and $A \supset [\bar{a}]\langle a\rangle A$.

Display calculi [2] can handle grammar logics with converse since they all fall into the primitive fragment identified by Kracht [21]. Display calculi all enjoy Belnap's general cut-elimination theorem, but it is well-known that they are not suitable for proof-search. Our work is motivated by the problem of automating proof search for display calculus. As in our previous work [11,12,13], we have chosen to work not directly in display calculus, but in a slightly different calculus based on nested sequents [19,4], which we call shallow nested sequent calculi. The syntactic constructs of nested sequents are closer to traditional sequent calculus, so as to allow us to use familiar notions in sequent calculus proof search procedures, such as the notions of saturation and loop checking, to automate proof search. A common feature of shallow nested sequent calculus and display calculus is the use of display postulates and other complex structural rules. These structural rules are the main obstacle to effective proof search, and our (proof theoretic) methodology for designing proof search calculi is guided by the problem of eliminating these structural rules entirely. We show here how our methodology can be used to derive proof search calculi for context-free grammar logics.

The general satisfiability problem for a grammar logic is to decide the satisfiability of a given formula when given a set of production rules for the underlying grammar. Nguyen and Szałas [23] give an excellent summary of what is known about this problem, as outlined next. Grammar logics were introduced by Fariñas del Cerro and Penttonen [8]. Baldoni et al [1] used prefixed tableaux to show that this problem is decidable for right linear logics but is undecidable for context free grammar logics. Demri [6] used an embedding into propositional dynamic logic with converse to prove this problem is EXPTIME-complete for right linear logics. Demri and de Nivelle [7] gave an embedding of the satisfiability problem for regular grammar logics into the two-variable guarded fragment of first-order logic and showed that satisfiability of regular grammar logics with converse is also EXPTIME-complete. Seen as description logics with inverse roles and complex role inclusions, decision procedures for regular grammar logics have also been studied extensively by Horrocks, et. al., see, e.g., [18,17,20]. Goré and Nguyen [10] gave an EXPTIME tableau decision procedure for the satisfiability of regular grammar logics using formulae labelled with automata states. Finally, Nguyen and Szałas [22,23] gave an extension of this method to handle converse by using the cut rule. In an unpublished manuscript, Nguyen has shown how to use the techniques of Goré and Widmann [15] to avoid the use of the cut rule. But as far as we know, there is no comprehensive sequent-style proof theory for grammar logics with converse which enjoys a syntactic cut-elimination theorem and which is amenable to proof-search.

We consider a proof theory, in nested sequent calculus, of grammar logics with converse, i.e., every modal operator $[a]$ comes with a converse $[\bar{a}]$. Extending previous works on nested sequent systems for (bi-)modal logics [11,13], we show, in Section 3, that all grammar logics (with or without converse) can be formalised in (shallow) nested sequent calculi, where the axioms are internalised in the calculi as structural rules. Syntactic cut-elimination for these calculi is proved using a procedure similar to that for display logics. We then show, in Section 4, that if the grammar is context-free, then one can get rid of all structural rules, in favor of *deep inference* [16,4] and propagation rules.

We then recast the problem of deciding grammar logics for the specific cases where the grammars are regular, using nested sequent calculus with deep inference. We first give, in Section 6.1, a decision procedure in the case where the regular grammar is given in the form of a FSA. This procedure is similar to existing tableaux-based decision procedures [17,22,23], where the states and transitions of the FSA are incorporated into proof rules for propagation of diamond-formulae. This procedure serves as a stepping stone to defining the more general decision procedure which does not depend on an explicit representation of axioms as a FSA in Section 6.2. The procedure in Section 6.2 is actually a semi-decision procedure that works on any finite set of context-free grammar axioms. However, we show that, in the case where the given grammar is regular, this procedure terminates. The procedure avoids the requirement to provide a FSA for the given axioms. This is significantly different from existing decision procedures for regular grammar logics [7,10,23,22], where it is assumed that a FSA encoding the axioms of the logics is given.

In this work, we follow Demri and de Nivelle's presentation of grammar axioms as a semi-Thue system [7]. The problem of deciding whether a context-free semi-Thue system is regular or not appears to be still open; see [20] for a discussion on this matter. Termination of our generic procedure for regular grammar logics of course does not imply solvability of this problem as it is dependent on the assumption that the given grammar is regular (see Theorem 6.11).

There are several proof systems to be presented next; we briefly describe the naming scheme used to differentiate these various sytems. We use Km to denote the Hilbert system for the basic multi-modal logic. Given a semi-Thue system S, we denote with Km(S) the extension of Km with axioms generated from S (see Section 2). There are two style of nested sequent calculus we consider here: the shallow-inference calculi (Section 3) and the deep-inference calculi (Section 4). Names of proof systems for the former are prefixed with an 'S', while names of proof systems for the latter are prefixed with a 'D'. For example, the corresponding shallow nested sequent calculus for Km (resp., Km(S)) is called SKm (resp., SKm(S)). In Section 5, we also consider shallow and deep nested sequent calculi parameterised by an automaton $\mathcal{A}$; these will be denoted by SKm($\mathcal{A}$) and DKm($\mathcal{A}$), respectively.

2 Grammar logics

The language of a multi-modal logic is defined w.r.t. to an alphabet Σ, used to index the modal operators. We use a, b and c, possibly with subscripts, for elements of Σ and use u and v, for elements of Σ^*, the set of finite strings over Σ. We use ϵ for the empty string. We define an operation $\bar{\cdot}$ (converse) on alphabets to capture converse modalities following Demri [7]. The converse operation satisfies $\bar{\bar{a}} = a$. We assume that Σ can be partitioned into two distinct sets Σ^+ and Σ^- such that $a \in \Sigma^+$ iff $\bar{a} \in \Sigma^-$. The converse operation is extended to strings in Σ^* as follows: if $u = a_1 a_2 \dots a_n$, then $\bar{u} = \bar{a}_n \bar{a}_{n-1} \dots \bar{a}_2 \bar{a}_1$, where $n \geq 0$. Note that if $u = \epsilon$ then $\bar{u} = \epsilon$.

We assume a given denumerable set of atomic formulae, ranged over by p, q, and r. The language of formulae is given by the following, where $a \in \Sigma$:

$$A ::= p \mid \neg A \mid A \vee A \mid A \wedge A \mid [a]A \mid \langle a \rangle A$$

Given a formula A, we write $A^\perp$ for the negation normal form (nnf) of $\neg A$. Implication $A \supset B$ is defined as $\neg A \vee B$.

Definition 2.1 *A* Σ-frame *is a pair $\langle W, R \rangle$ of a non-empty set W of worlds and a set R of binary relations $\{R_a\}_{a \in \Sigma}$ over W satisfying, for every $a \in \Sigma$, $R_a = \{(x, y) \mid R_{\bar{a}}(y, x)\}$. A* valuation *$V$ is a mapping from propositional variables to sets of worlds. A* model *$\mathfrak{M}$ is a triple $\langle W, R, V \rangle$ where $\langle W, R \rangle$ is a frame and V is a valuation. The relation $\models$ is defined inductively as follows:*

- $\mathfrak{M}, x \models p$ *iff* $x \in V(p)$.
- $\mathfrak{M}, x \models \neg A$ *iff* $\mathfrak{M}, x \not\models A$.
- $\mathfrak{M}, x \models A \wedge B$ *iff* $\mathfrak{M}, x \models A$ *and* $\mathfrak{M}, x \models B$.
- $\mathfrak{M}, x \models A \vee B$ *iff* $\mathfrak{M}, x \models A$ *or* $\mathfrak{M}, x \models B$.
- *For every $a \in \Sigma$, $\mathfrak{M}, x \models [a]A$ iff for every y such that $R_a(x, y)$, $\mathfrak{M}, y \models A$.*
- *For every $a \in \Sigma$, $\mathfrak{M}, x \models \langle a \rangle A$ iff there exists y such that $R_a(x, y)$, $\mathfrak{M}, y \models A$.*

A formula A is satisfiable *iff there exists a Σ-model $\mathfrak{M} = \langle W, R, V \rangle$ and a world $x \in W$ such that $\mathfrak{M}, x \models A$.*

We now define a class of multi-modal logics, given Σ, that is induced by *production rules* for strings from Σ^*. We follow the framework in [7], using semi-Thue systems to define the logics. A production rule is a binary relation over strings in Σ^*, interpreted as a rewrite rule on strings. We use the notation $u \to v$ to denote a production rule which rewrites u to v. A *semi-Thue* system is a set S of production rules. It is *closed* if $u \to v \in S$ implies $\bar{u} \to \bar{v} \in S$.

Given a Σ-frame $\langle W, R \rangle$, we define another family of accessibility relations indexed by Σ^* as follows: $R_\epsilon = \{(x, x) \mid x \in W\}$ and for every $u \in \Sigma^*$ and for every $a \in \Sigma$, $R_{ua} = \{(x, y) \mid (x, z) \in R_u, (z, y) \in R_a, \text{ for some } z \in W\}$.

Definition 2.2 *Let $u \to v$ be a production rule and let $\mathcal{F} = \langle W, R \rangle$ be a Σ-frame. $\mathcal{F}$ is said to satisfy $u \to v$ if $R_v \subseteq R_u$. $\mathcal{F}$ satisfies a semi-Thue system S if it satisfies every production rule in S.*

Definition 2.3 *Let S be a semi-Thue system. A formula A is said to be S-*satisfiable *iff there is a model $\mathfrak{M} = \langle W, R, V\rangle$ such that $\langle W, R\rangle$ satisfies S and $\mathfrak{M}, x \models A$ for some $x \in W$. A is said to be S-*valid *if for every Σ-model $\mathfrak{M} = \langle W, R, V\rangle$ that satisfies S, we have $\mathfrak{M}, x \models A$ for every $x \in W$.*

Given a string $u = a_1 a_2 \ldots a_n$ and a formula A, we write $\langle u\rangle A$ for the formula $\langle a_1\rangle\langle a_2\rangle \cdots \langle a_n\rangle A$. The notation $[u]A$ is defined analogously. If $u = \epsilon$ then $\langle u\rangle A = [u]A = A$.

Definition 2.4 *Let S be a closed semi-Thue system over an alphabet Σ. The system* Km(S) *is an extension of the standard Hilbert system for multi-modal* Km *(see, e.g., [3]) with the following axioms:*

- *for each $a \in \Sigma$, a* residuation axiom: $A \supset [a]\langle \bar{a}\rangle A$
- *and for each $u \to v \in S$, an axiom $[u]A \supset [v]A$.*

Because S is closed, each axiom $[u]A \supset [v]A$ has an *inverted version* $[\bar{u}]A \supset [\bar{v}]A$. The following theorem can be proved following a similar soundness and completeness proof for Hilbert systems for modal logics (see, e.g., [3]).

Theorem 2.5 *A formula F is S-valid iff F is provable in* Km(S).

3 Nested sequent calculi with shallow inference

We now give a sequent calculus SKm for Km(S), by using the framework of nested sequent calculus [19,4,11,13]. We follow the notation used in [19,13], extended to the multi-modal case. From this section onwards, we shall be concerned only with formulae in nnf, so we can restrict to one-sided sequents.

A nested sequent is a multiset of the form shown below at left where each A_i is a formula and each Δ_i is a nested sequent:

$$A_1, \ldots, A_m, (a_1)\{\Delta_1\}, \ldots, (a_n)\{\Delta_n\} \qquad\qquad A_1 \vee \cdots \vee A_m \vee [a_1]B_1 \vee \cdots \vee [a_n]B_n$$

We shall often drop the outermost braces when writing nested sequents, e.g., writing Γ, Δ instead of $\{\Gamma, \Delta\}$. The structural connective $(a)\{.\}$ is a proxy for the modality $[a]$, so this nested sequent can be interpreted as the formula shown above right (modulo associativity and commutativity of $\vee$), where each B_i is the interpretation of Δ_i. We shall write $(u)\{\Delta\}$, where $u = a_1 \cdots a_n \in \Sigma^*$, to denote the structure:

$$(a_1)\{(a_2)\{\cdots (a_n)\{\Delta\}\} \cdots\}.$$

A *context* is a nested sequent with a 'hole' [ ] in place of a formula: this notation should not be confused with the modality $[a]$. We use $\Gamma[\ ]$, $\Delta[\ ]$, etc. for contexts. Given a context $\Gamma[\ ]$ and a nested sequent Δ, we write $\Gamma[\Delta]$ to denote the nested sequent obtained by replacing the hole in $\Gamma[\ ]$ with Δ.

The core inference rules for multi-modal SKm (without axioms) are given in Figure 1. The rule r is called a *residuation rule* (or display postulate) and corresponds to the residuation axioms.

$$\frac{}{\Gamma, p, \neg p}\ id \qquad \frac{\Gamma, A \quad \Delta, A^{\perp}}{\Gamma, \Delta}\ cut \qquad \frac{\Gamma, \Delta, \Delta}{\Gamma, \Delta}\ ctr \qquad \frac{\Gamma}{\Gamma, \Delta}\ wk \qquad \frac{\Gamma, (a)\{\Delta\}}{(\bar{a})\{\Gamma\}, \Delta}\ r$$

$$\frac{\Gamma, A \quad \Gamma, B}{\Gamma, A \wedge B}\ \wedge \qquad \frac{\Gamma, A, B}{\Gamma, A \vee B}\ \vee \qquad \frac{\Gamma, (a)\{A\}}{\Gamma, [a]A}\ [a] \qquad \frac{\Gamma, (a)\{\Delta, A\}}{\Gamma, (a)\{\Delta\}, \langle a \rangle A}\ \langle a \rangle$$

Fig. 1. The inference rules of the shallow nested sequent calculus SKm

To capture Km(S), we need to convert each axiom generated from S to an inference rule. Each production rule $u \to v$ gives rise to the axiom $[u]A \supset [v]A$, or equivalently, $\langle \bar{v} \rangle A \supset \langle \bar{u} \rangle A$. The latter is an instance of Kracht's *primitive axioms* [21] (generalised to the multimodal case). Thus, we can convert the axiom into a structural rule following Kracht's rule scheme for primitive axioms:

$$\frac{(u)\{\Delta\}, \Gamma}{(v)\{\Delta\}, \Gamma}$$

Let $\rho(S)$ be the set of structural rules induced by the semi-Thue system S.

Definition 3.1 *Let S be a closed semi-Thue system S over an alphabet Σ.* SKm(S) *is the proof system obtained by extending* SKm *with* $\rho(S)$.

We say that two proof systems are equivalent if and only if they prove the same set of formulae.

Theorem 3.2 *The system* SKm(S) *and* Km(S) *are equivalent.*

Proof. *(Outline).* In one direction, from SKm(S) to Km(S), we show that, for each inference rule of SKm(S), if the formula interpretation of the premise(s) is valid then the formula interpretation of the conclusion is also valid. For the converse, it is enough to show that all axioms of Km(S) are derivable in SKm(S). It can be shown that both the residuation axioms and the axioms generated from S can be derived using the structural rules r and $\rho(S)$. □

The cut-elimination proof for SKm(S) follows a similar generic procedure for display calculi [2,21], which has been adapted to nested sequent in [13]. The key to cut-elimination is to show that SKm(S) has the *display property.*

Lemma 3.3 *Let $\Gamma[\Delta]$ be a nested sequent. Then there exists a nested sequent Γ' such that $\Gamma[\Delta]$ is derivable from the nested sequent Γ', Δ, and vice versa, using only the residuation rule r. Intuitively, the nested Δ in $\Gamma[\Delta]$ is displayed at the top level in Γ', Δ.*

Theorem 3.4 *Cut elimination holds for* SKm(S).

Proof. This is a straightforward adaptation of the cut-elimination proof in [13] for tense logic. □

4 Deep inference calculi

Although the shallow system SKm(S) enjoys cut-elimination, proof search in its cut-free fragment is difficult to automate, due to the presence of structural

$$\frac{}{\Gamma[p, \neg p]}\ id_d \qquad \frac{\Gamma[A \wedge B, A] \quad \Gamma[A \wedge B, B]}{\Gamma[A \wedge B]}\ \wedge_d \qquad \frac{\Gamma[A \vee B, A, B]}{\Gamma[A \vee B]}\ \vee_d$$

$$\frac{\Gamma[[a]A, (a)\{A\}]}{\Gamma[[a]A]}\ [a]_d \qquad \frac{\Gamma[(a)\{\Delta, A\}, \langle a \rangle A]}{\Gamma[(a)\{\Delta\}, \langle a \rangle A]}\ \langle a{\uparrow} \rangle \qquad \frac{\Gamma[(a)\{\Delta, \langle \bar{a} \rangle A\}, A]}{\Gamma[(a)\{\Delta, \langle \bar{a} \rangle A\}]}\ \langle a{\downarrow} \rangle$$

Fig. 2. The inference rules of DKm

rules. To reduce the non-determinism caused by structural rules, we consider next a proof system in which all structural rules (including those induced by grammar axioms) can be absorbed into logical rules. As the display property in Lemma 3.3 suggests, the residuation rule allows one to essentially apply an inference rule to a particular subsequent nested inside a nested sequent, by displaying that subsequent at the top level and undisplaying it back to its original position in the nested sequent. It is therefore quite intuitive that one way to get rid of the residuation rule is to allow *deep inference* rules, that apply deeply within any arbitrary context in a nested sequent.

The deep inference system DKm, which corresponds to SKm, is given in Figure 2. As can be readily seen, the residuation rule is absent and contraction and weakening are absorbed into logical rules.

To fully absorb the residuation rule and other structural rules induced by semi-Thue systems, we need to introduce additional introduction rules for the diamond operator $\langle a \rangle$, which we call *propagation rules.* We shall show next how these propagation rules are generated from a semi-Thue system.

Let S be a closed semi-Thue system over alphabet Σ. We write $u \Rightarrow_S v$ to mean that the string v can be reached from u by applying the production rules (as rewrite rules) in S successively to u. Define $L_a(S) = \{u \mid a \Rightarrow_S u\}$. Then $L_a(S)$ defines a language generated from S with the start symbol a.

A nested sequent can be seen as a tree whose nodes are multisets of formulae, and whose edges are labeled with elements of Σ. We assume that each node in a nested sequent can be identified uniquely, i.e., we can consider each node as labeled with a unique position identifier. An internal node of a nested sequent is a node which is not a leaf node. We write $\Gamma[\ ]_i$ to denote a context in which the hole is located in the node at position i in the tree representing $\Gamma[\ ]$. This generalises to multi-contexts, so $\Gamma[\ ]_i[\ ]_j$ denotes a two-hole context, one hole located at i and the other at j (they can be the same location). From now on, we shall often identify a nested sequent with its tree representation, so when we speak of a node in Γ, we mean a node in the tree of Γ. If i and j are nodes in Γ, we write $i \succ^a j$ when j is a child node of i and the edge from i to j is labeled with a. If i is a node in the tree of Γ, we write $\Gamma|i$ to denote the multiset of formula occuring in the node i. Let Δ and Γ be nested sequents. Suppose i is a node in Γ. Then we write $\Gamma(i \ll \Delta)$ for the nested sequent obtained from Γ by adding Δ to node i in Γ. Note that for such an addition to preserve the uniqueness of the position identifiers of the resulting tree, we need to rename the identifiers in Δ to avoid clashes. We shall assume implicitly that such a renaming is carried out when we perform this addition.

Definition 4.1 (Propagation automaton.) *A propagation automaton is a finite state automaton* $\mathcal{P} = (\Sigma, Q, I, F, \delta)$ *where* Q *is a finite set of states,* $I = \{s\}$ *is a singleton set of an initial state and* $F = \{t\}$ *is a singleton set of a final state with* $s, t \in Q$*, and for every* $i, j \in Q$*, if* $i \xrightarrow{a} j \in \delta$ *then* $j \xrightarrow{\bar{a}} i \in \delta$*.*

In other words, a propagation automaton is just a finite state automaton (FSA) where each transition has a dual transition.

Definition 4.2 *Let* $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ *be a FSA. Let* $\boldsymbol{i} = i_1, \dots, i_n$ *and* $\boldsymbol{j} = j_1, \dots, j_n$ *be two sequences of states in* Q*. Let* $[i_1 := j_1, \dots, i_n := j_n]$ *(we shall abbreviate this as* $[\boldsymbol{i} := \boldsymbol{j}]$*) be a (postfix) mapping from* Q *to* Q *that maps* i_m *to* j_m*, where* $1 \leq m \leq n$*, and is the identity map otherwise. This mapping is extended to a (postfix) mapping between sets of states as follows: given* $Q' \subseteq Q$*,* $Q'[\boldsymbol{i} := \boldsymbol{j}] = \{k[\boldsymbol{i} := \boldsymbol{j}] \mid k \in Q'\}$*. The automaton* $\mathcal{A}[\boldsymbol{i} := \boldsymbol{j}]$ *is the tuple* $(\Sigma, Q[\boldsymbol{i} := \boldsymbol{j}], I[\boldsymbol{i} := \boldsymbol{j}], F[\boldsymbol{i} := \boldsymbol{j}], \delta')$ *where*

$$\delta' = \{k[\boldsymbol{i} := \boldsymbol{j}] \xrightarrow{a} l[\boldsymbol{i} := \boldsymbol{j}] \mid k \xrightarrow{a} l \in \delta\}.$$

To each nested sequent Γ, and nodes i and j in Γ, we associate a propagation automaton $\mathcal{R}(\Gamma, i, j)$ as follows:

(i) the states of $\mathcal{R}(\Gamma, i, j)$ are the nodes of (the tree of) Γ;

(ii) i is the initial state of $\mathcal{R}(\Gamma, i, j)$ and j is its final state;

(iii) each edge $x \succ^a y$ in Γ corresponds to two transitions in $\mathcal{R}(\Gamma, i, j)$: namely, $x \xrightarrow{a} y$ and $y \xrightarrow{\bar{a}} x$.

Note that although propagation automata are defined for nested sequents, they can be similarly defined for (multi-)contexts as well, as contexts are just sequents containing a special symbol $[\ ]$ denoting a hole. So in the following, we shall often treat a context as though it is a nested sequent.

A semi-Thue system S over alphabet Σ is *context-free* if its production rules are all of the form $a \to u$ for some $a \in \Sigma$.

In the following, to simplify presentation, we shall use the same notation to refer to an automaton $\mathcal{A}$ and the regular language it accepts. Given a context-free closed semi-Thue system S, the *propagation rules for* S are all the rules of the following form where i and j are two (not necessarily distinct) nodes of Γ:

$$\frac{\Gamma[\langle a \rangle A]_i [A]_j}{\Gamma[\langle a \rangle A]_i [\emptyset]_j}\ p_S, \text{ provided } \mathcal{R}(\Gamma[\ ]_i[\ ]_j, i, j) \cap L_a(S) \neq \emptyset.$$

Note that the intersection of a regular language and a context-free language is a context-free language (see, e.g., Chapter 3 in [9] for a construction of the intersection), and since the problem of checking emptiness for context-free languages is decidable [9], the rule p_S can be effectively mechanised.

Definition 4.3 *Given a context-free closed semi-Thue system S over an alphabet Σ, the proof system* $\mathrm{DKm}(S)$ *is obtained by extending* DKm *with* p_S.

We now show that $\mathrm{DKm}(S)$ is equivalent to $\mathrm{SKm}(S)$. The proof relies on a series of lemmas showing admissibility of all structural rules of $\mathrm{SKm}(S)$ in $\mathrm{DKm}(S)$. The proof follows the same outline as in the case for tense logic [13]. The adaptation of the proof in [13] is quite straightforward, so we shall not go into detailed proofs but instead just outline the required lemmas. Some of their proofs are outlined in the appendix. In the following lemmas, we shall assume that S is a closed context-free semi-Thue system over some Σ.

Given a derivation Π, we denote with $|\Pi|$ the *height* of Π, which is simply the length (i.e., the number of edges) of the longest branch in Π. A single premise rule ρ is said to be *admissible* in $\mathrm{DKm}(S)$ if provability of its premise in $\mathrm{DKm}(S)$ implies provability of its conclusion in $\mathrm{DKm}(S)$. It is *height-preserving admissible* if whenever the premise has a derivation then the conclusion has a derivation of the same height, in $\mathrm{DKm}(S)$.

Admissibility of the weakening rule is a consequence of the following lemma.

Lemma 4.4 *Let Π be a derivation of $\Gamma[\emptyset]$ in $\mathrm{DKm}(S)$. Then, for all nested sequents Δ, there exists a derivation Π' of $\Gamma[\Delta]$ in $\mathrm{DKm}(S)$ such that $|\Pi| = |\Pi'|$.*

The admissibility proofs of the remaining structural rules all follow the same pattern: the most important property to prove is that, if a propagation path for a diamond formula exists between two nodes in the premise, then there exists a propagation path for the same formula, between the same nodes, in the conclusion of the rule.

Lemma 4.5 *The rule r is height-preserving admissible in* $\mathrm{DKm}(S)$.

Admissibility of contraction is proved indirectly by showing that it can be replaced by a formula contraction rule and a distributivity rule:

$$\frac{\Gamma[A, A]}{\Gamma[A]}\ actr \qquad \frac{\Gamma[(a)\{\Delta_1\}, (a)\{\Delta_2\}]}{\Gamma[(a)\{\Delta_1, \Delta_2\}]}\ m$$

The rule m is also called a *medial* rule and is typically used to show admissibility of contraction in deep inference [5].

Lemma 4.6 *The rule ctr is admissible in* $\mathrm{DKm}(S)$ *plus actr and m.*

Lemma 4.7 *The rules actr and m are height-preserving admissible in* $\mathrm{DKm}(S)$.

Admissibility of contraction then follows immediately.

Lemma 4.8 *The contraction rule ctr is admissible in* $\mathrm{DKm}(S)$.

Lemma 4.9 *The structural rules $\rho(S)$ of* $\mathrm{SKm}(S)$ *are height-preserving admissible in* $\mathrm{DKm}(S)$.

Theorem 4.10 *For every context-free closed semi-Thue system S, the proof systems* $\mathrm{SKm}(S)$ *and* $\mathrm{DKm}(S)$ *are equivalent.*

5 Regular grammar logics

A context free semi-Thue system S over Σ is regular if for every $a \in \Sigma$, the language $L_a(S)$ is a regular language. In this section, we consider logics generated by regular closed semi-Thue systems. We assume in this case that the union of the regular languages $\{L_a(S) \mid a \in \Sigma\}$ is represented explicitly as an FSA $\mathcal{A}$ with no silent transitions. Thus $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and δ is the transition relation. Given $\mathcal{A}$ as above, we write $s \xrightarrow{a}_{\mathcal{A}} t$ to mean $s \xrightarrow{a} t \in \delta$. We assume that each $a \in \Sigma$ has a unique initial state $init_a \in I$.

We shall now define an alternative deep inference system given this explicit representation of the grammar axioms as an FSA. Following similar tableaux systems in the literature that utilise such an automaton representation [17,22,23], we use the states of the FSA to index formulae in a nested sequent to record stages of propagation. For this, we first introduce a form of labeled formula, written $s : A$, where $s \in Q$. The propagation rules corresponding to $\mathcal{A}$ are:

$$\frac{\Gamma[\langle a\rangle A, init_a : A]}{\Gamma[\langle a\rangle A]}\ i \qquad \frac{\Gamma[s : A, (a)\{s' : A, \Delta\}]}{\Gamma[s : A, (a)\{\Delta\}]}\ t\uparrow, \text{ if } s \xrightarrow{a}_{\mathcal{A}} s'$$

$$\frac{\Gamma[s : A, A]}{\Gamma[s : A]}\ f, \text{ if } s \in F \qquad \frac{\Gamma[(a)\{s : A, \Delta\}, s' : A]}{\Gamma[(a)\{s : A, \Delta\}]}\ t\downarrow, \text{ if } s \xrightarrow{\bar{a}}_{\mathcal{A}} s'.$$

Definition 5.1 *Let S be a regular closed semi-Thue system over Σ and let $\mathcal{A}$ be an FSA representing the regular language generated by S and Σ.* DKm($\mathcal{A}$) *is the proof system* DKm *extended with the rules $\{i, f, t\downarrow, t\uparrow\}$ for $\mathcal{A}$.*

It is intuitively clear that DKm($\mathcal{A}$) and DKm(S) are equivalent, when $\mathcal{A}$ defines the same language as $L(S)$. Essentially, a propagation rule in DKm(S) can be simulated by DKm($\mathcal{A}$) using one or more propagations of labeled formulae. The other direction follows from the fact that when a diamond formula $\langle a\rangle A$ is propagated, via the use of labeled formulae, to a labeled formula $s : A$ where s is a final state, then there must be a chain of transitions between labeled formulae for A whose string forms an element of $\mathcal{A}$, hence also in $L_a(S)$. One can then propagate directly $\langle a\rangle A$ in DKm(S).

Theorem 5.2 *Let S be a regular closed semi-Thue system over Σ and let $\mathcal{A}$ be a FSA representing the regular language generated by S and Σ. Then* DKm(S) *and* DKm($\mathcal{A}$) *are equivalent.*

6 Decision procedures

We now show how the proof systems DKm($\mathcal{A}$) and DKm(S) can be turned into decision procedures for regular grammar logics. Our aim is to derive the decision procedure for DKm(S) directly without the need to convert S explicitly to an automaton; the decision procedure DKm($\mathcal{A}$) will serve as a stepping stone towards this aim. The decision procedure for DKm(S) is a departure from all existing decision procedures for regular grammar logics (with or without converse) [17,7,10,22,23] that assume that an FSA representing S is given.

$Prove_1(\mathcal{A}, \Gamma)$

(i) If $\Gamma = \Gamma'[p, \neg p]$, return $\top$.

(ii) If Γ is $\mathcal{A}$-stable, return $\bot$.

(iii) If Γ is not saturated:
   - (a) If $A \vee B \in \Gamma|i$ but $A \notin \Gamma|i$ or $B \notin \Gamma|i$, then let $\Gamma' := \Gamma(i \ll \{A, B\})$ and return $Prove_1(\mathcal{A}, \Gamma')$.
   - (b) Suppose $A_1 \wedge A_2 \in \Gamma|i$ but neither $A_1 \in \Gamma|i$ nor $A_2 \in \Gamma|i$. Let $\Gamma_1 = \Gamma(i \ll \{A_1\})$ and $\Gamma_2 = \Gamma(i \ll \{A_2\})$. Then return $\bot$ if $Prove_1(\mathcal{A}, \Gamma_j) = \bot$ for some $j \in \{1, 2\}$. Otherwise return $\top$.

(iv) If Γ is not $\mathcal{A}$-propagated: then there is a node i s.t. one of the following applies:
   - (a) $\langle a \rangle A \in \Gamma|i$ but $init_a : A \notin \Gamma|i$. Then let $\Gamma' := \Gamma(i \ll \{init_a : A\})$.
   - (b) $s : A \in \Gamma|i$ and $s \in F$, but $A \notin \Gamma|i$. Then let $\Gamma' := \Gamma(i \ll \{A\})$.
   - (c) $s : A \in \Gamma|i$, there is j s.t. $i \succ^a j$ and $s \xrightarrow{a}_{\mathcal{A}} t$, but $t : A \notin \Gamma|j$. Then let $\Gamma' := \Gamma(j \ll \{t : A\})$.
   - (d) $s : A \in \Gamma|i$, there is j s.t. $j \succ^a i$ and $s \xrightarrow{\bar{a}}_{\mathcal{A}} t$, but $t : A \notin \Gamma|j$. Then let $\Gamma' := \Gamma(j \ll \{t : A\})$.

   Return $Prove_1(\mathcal{A}, \Gamma')$.

(v) If there is an internal node i in Γ that is not realised: Then there is $[a]A \in \Gamma|i$ such that $A \notin \Gamma|j$ for every j s.t. $i \succ^a j$. Let $\Gamma' := \Gamma(i \ll (a)\{A\})$. Return $Prove_1(\mathcal{A}, \Gamma')$.

(vi) If there is a leaf node i that is not realised and is not a loop node: Then there is $[a]A \in \Gamma|i$. Let $\Gamma' := \Gamma(i \ll (a)\{A\})$. Return $Prove_1(\mathcal{A}, \Gamma')$.

Fig. 3. An automata-based prove procedure.

6.1 An automata-based procedure

The decision procedure for DKm($\mathcal{A}$) is basically just backward proof search, where one tries to saturate each sequent in the tree of sequents until either the id_d rule is applicable, or a certain stable state is reached. When the latter is reached, we show that a counter model to the original nested sequent can be constructed. Although we obtain this procedure via a different route, the end result is very similar to the tableaux-based decision procedure in [17]. In particular, our notion of a stable state (see Definition 6.3) used to block proof search is the same as the blocking condition in tableaux systems [17,7,10,23,22], which takes advantange of the labeling of formulae with the states of the automaton.

Recall the notation $\Gamma|i$ refers to the multiset of formulae at node i in Γ.

Definition 6.1 (Saturation and realisation) *A node i in Γ is* saturated *if the following hold:*

(i) *If $A \in \Gamma|i$ then $A^\perp \notin \Gamma|i$.*

(ii) *If $A \vee B \in \Gamma|i$ then $A \in \Gamma|i$ and $B \in \Gamma|i$.*

(iii) *If $A \wedge B \in \Gamma|i$ then $A \in \Gamma|i$ or $B \in \Gamma|i$.*

$\Gamma|i$ *is* realised *if* $[a]A \in \Gamma|i$ *implies that there exists* j *such that* $i \succ^a j$ *and* $A \in \Gamma|j$.

Definition 6.2 ($\mathcal{A}$-propagation) *Let* $\mathcal{A} = (\Sigma, Q, I, F, \delta)$. *A nested sequent* Γ *is said to be* $\mathcal{A}$*-propagated if for every node* i *in* Γ, *the following hold:*

(i) *If* $\langle a \rangle A \in \Gamma|i$ *then* $init_a : A \in \Gamma|i$ *for any* $a \in \Sigma$.

(ii) *If* $s : A \in \Gamma|i$ *and* $s \in F$, *then* $A \in \Gamma|i$.

(iii) *For all* j, a, s *and* t, *such that* $i \succ^a j$ *and* $s \xrightarrow{a}_{\mathcal{A}} t$, *if* $s : A \in \Gamma|i$ *then* $t : A \in \Gamma|j$.

(iv) *For all* j, a, s *and* t, *such that* $j \succ^a i$ *and* $s \xrightarrow{\bar{a}}_{\mathcal{A}} t$, *if* $s : A \in \Gamma|i$ *then* $t : A \in \Gamma|j$.

Definition 6.3 ($\mathcal{A}$-stability) *A nested sequent* Γ *is* $\mathcal{A}$-stable *if*

(i) *Every node is saturated.*

(ii) Γ *is* $\mathcal{A}$*-propagated.*

(iii) *Every internal node is realised.*

(iv) *For every leaf node* i, *one of the following holds:*
   (a) *There is an ancestor node* j *of* i *such that* $\Gamma|i = \Gamma|j$. *We call the node* i *a* loop node.
   (b) $\Gamma|i$ *is realised (i.e., it cannot have a member of the form* $[a]A$*).*

The prove procedure for DKm($\mathcal{A}$) is given in Figure 3. We show that the procedure is sound and complete with respect to DKm($\mathcal{A}$). The proofs of the following theorems can be found in the appendix.

Theorem 6.4 *If* $Prove_1(\mathcal{A}, \{F\})$ *returns* $\top$ *then* F *is provable in* DKm($\mathcal{A}$). *If* $Prove_1(\mathcal{A}, \{F\})$ *returns* $\bot$ *then* F *is not provable in* DKm($\mathcal{A}$).

Theorem 6.5 *For every formula* A, $Prove_1(\mathcal{A}, \{A\})$ *terminates.*

Corollary 6.6 *The proof system* DKm($\mathcal{A}$) *is decidable.*

6.2 A grammar-based procedure

The grammar-based procedure differs from the automaton-based procedure in the notion of propagation and that of a stable nested sequent.

In the following, given a function θ from labels to labels, and a list $\boldsymbol{i} = i_1, \dots, i_n$ of labels, we write $\theta(\boldsymbol{i})$ to denote the list $\theta(i_1), \dots, \theta(i_n)$. We write $[\boldsymbol{i} := \theta(\boldsymbol{i})]$ to mean the mapping $[i_1 := \theta(i_1), \dots, i_n := \theta(i_n)]$.

In the following definitions, S is assumed to be a context-free semi-Thue system over some alphabet Σ.

Definition 6.7 (S-propagation) *Let* Γ *be a nested sequent. Let* $\mathcal{P} = (\Sigma, Q, \{i\}, \{j\}, \delta)$ *be a propagation automata, where* Q *is a subset of the nodes in* Γ. *We say that* Γ *is* $(S, \mathcal{P})$-propagated *if the following holds:* $\langle a \rangle A \in \Gamma|i$ *and* $\mathcal{P} \cap L_a(S) \neq \emptyset$ *imply* $A \in \Gamma|j$. Γ *is* S-propagated *if it is* $(S, \mathcal{R}(\Gamma, i, j))$*-propagated for every node* i *and* j *in* Γ.

Definition 6.8 (S-stability) *A nested sequent* Γ *is* S-stable *if*

```
$Prove_2(S, \Gamma, k)$
  (i) If $\Gamma = \Gamma'[p, \neg p]$, return $\top$.
  (ii) If $\Gamma$ is $S$-stable, return $\bot$.
  (iii) If $\Gamma$ is not saturated:
      • If $A \vee B \in \Gamma|i$ but $A \notin \Gamma|i$ or $B \notin \Gamma|i$, then let $\Gamma' := \Gamma(i \ll \{A, B\})$ and
        return $Prove_2(S, \Gamma', k)$.
      • Suppose $A_1 \wedge A_2 \in \Gamma|i$ but neither $A_1 \in \Gamma|i$ nor $A_2 \in \Gamma|i$. Let $\Gamma_1 =$
        $\Gamma(i \ll \{A_1\})$ and $\Gamma_2 = \Gamma(i \ll \{A_2\})$. If $Prove_2(S, \Gamma_j, k) = \bot$, for some
        $j \in \{1, 2\}$, then return $\bot$. Otherwise, if $Prove_2(S, \Gamma_j, k) = \star$ for some
        $j \in \{1, 2\}$, then return $\star$. If $Prove_2(S, \Gamma_j, k) = \top$, for all $j \in \{1, 2\}$, then
        return $\top$.
  (iv) If $\Gamma$ is not $S$-propagated: then there must be nodes $i$ and $j$ such that
       $\langle a \rangle A \in \Gamma|i$ and $\mathcal{R}(\Gamma, i, j) \cap L_a(S) \neq \emptyset$, but $A \notin \Gamma|j$. Let $\Gamma' := \Gamma(j \ll \{A\})$.
       Return $Prove_2(S, \Gamma', k)$.
  (v) If there is an internal node $i$ in $\Gamma$ that is not realised: Then there is $[a]A \in$
      $\Gamma|i$ such that $A \notin \Gamma|j$ for every $j$ s.t. $i \succ^a j$. Let $\Gamma' := \Gamma(i \ll (a)\{A\})$.
      Return $Prove_2(S, \Gamma', k)$.
  (vi) Non-deterministically choose a leaf node $i$ that is not realised and is at
       height equal to or lower than $k$ in $\Gamma$: Then there is $[a]A \in \Gamma|i$. Let
       $\Gamma' := \Gamma(i \ll (a)\{A\})$. Return $Prove_2(S, \Gamma', k)$.
  (vii) Return $\star$.

$Prove(S, \Gamma)$
  (i) $k := 0$.
  (ii) If $Prove_2(S, \Gamma, k) = \top$ or $Prove_2(S, \Gamma, k) = \bot$, return $\top$ or $\bot$ respectively.
  (iii) $k := k + 1$. Go to step (ii).
```

Fig. 4. A grammar-based prove procedure.

(i) *Every node is saturated.*

(ii) Γ *is S-propagated.*

(iii) *Every internal node is realised.*

(iv) *Let $\boldsymbol{x} = x_1, \ldots, x_n$ be the list of all unrealised leaf nodes. There is a function λ assigning each unrealised leaf node x_m to an ancestor $\lambda(x_m)$ of x_m such that $\Gamma|x_m = \Gamma|\lambda(x_m)$ and for every node y and z, Γ is $(S, \mathcal{P})$-propagated, where $\mathcal{P} = \mathcal{R}(\Gamma, y, z)[\boldsymbol{x} := \lambda(\boldsymbol{x})]$.*

Now we define a non-deterministic prove procedure $Prove_2(S, \Gamma, k)$ as in Figure 4, where k is an integer and S is a context-free closed semi-Thue system. Given a nested sequent Γ, and a node i in Γ, the *height* of i in Γ is the length of the branch from the root of Γ to node i. The procedure $Prove_2(S, \Gamma, k)$ tries to construct a derivation of Γ, but is limited to exploring only those nested sequents derived from Γ that has height at most k. The procedure *Prove* given below is essentially an iterative deepening procedure that calls $Prove_2$ repeatedly with increasing values of k. If an input sequent is not valid, the procedure will try to guess the smallest S-stable sequent that refutes the input sequent, i.e., it essentially tries to construct a finite countermodel. The procedure *Prove* gives a semi-decision procedure for context-free grammar logics. This uses the following lemma about S-stable sequents, which shows how to extract a countermodel from an S-stable sequent.

Lemma 6.9 *Let S be a context-free closed semi-Thue system. If Γ is an S-stable nested sequent, then there exists a model $\mathfrak{M}$ such that for every node x in Γ, there exists a world w in $\mathfrak{M}$ such that for every $A \in \Gamma|x$, we have $\mathfrak{M}, w \not\models A$.*

Theorem 6.10 *Let S be a context-free closed semi-Thue system. For every formula F, $Prove(S, \{F\})$ returns $\top$ if and only if F is provable in* $\mathrm{DKm}(S)$.

We next show that $Prove(S, \Gamma)$ terminates when S is regular. The key is to bound the size of S-stable sequents, hence the non-deterministic iterative deepening will eventually find an S-stable sequent, when Γ is not provable.

Theorem 6.11 *Let S be a regular closed semi-Thue system over an alphabet Σ. Then for every formula F, the procedure $Prove(S, \{F\})$ terminates.*

The proof relies on the fact that there exists a minimal FSA $\mathcal{A}$ encoding S, so one can simulate steps of $Prove_1(\mathcal{A}, \{F\})$ in $Prove(S, \{F\})$. It is not difficult to show that if a run of $Prove_1(\mathcal{A}, \{F\})$ reaches an $\mathcal{A}$-stable nested sequent Γ', then one can find a k such that a run of $Prove_2(S, \{F\}, k)$ reaches a saturated and S-propagated nested sequent Δ, such that Γ' and Δ are identical except for the labeled formulae in Γ'. The interesting part is in showing that Δ is S-stable. The details are in the appendix.

The following is then a corollary of Theorem 6.10 and Theorem 6.11.

Corollary 6.12 *Let S be a regular closed semi-Thue system over an alphabet Σ. Then the procedure Prove is a decision procedure for* $\mathrm{DKm}(S)$.

7 Conclusion and future work

Nested sequent calculus is closely related to display calculi, allowing us to benefit from well-studied proof theoretic techniques in display calculi, such as Belnap's generic cut-elimination procedure, to prove cut-elimination for $\mathrm{SKm}(S)$. At the more practical end, we have established via proof theoretic means that nested sequent calculi for regular grammar logics can be effectively mechanised. This work and our previous work [11,13] suggests that nested sequent calculus could potentially be a good intermediate framework to study both proof theory and decision procedures, at least for modal and substructural logics.

Nested sequent calculus can be seen as a special case of labelled sequent calculus, as the tree structure in a nested sequent can be encoded using labels and accessibility relations among these labels in labelled calculi. The relation between the two has recently been established in [24], where the author shows that, if one gets rid of the frame rules in labelled calculi and structural rules in nested sequent calculi, there is a direct mapping between derivations of formulae between the two frameworks. However, it seems that the key to this connection, i.e., admissibility of the frame rules, has already been established in Simpson's thesis [25], where he shows admissibility of a class of frame rules in favour of propagation rules obtained by applying a closure operation on these frame rules. The latter is similar to our notion of propagation rules. Thus it seems that structural rules in (shallow) nested sequent calculus play a role similar to the frame rules in labelled calculi. We plan to investigate this connection further, e.g., under what conditions are the structural rules admissible in deep inference calculi, and whether those conditions translate into any meaningful characterisations in terms of (first-order) properties of frames.

The two decision procedures for regular grammar logics we have presented are not optimal. As can be seen from the termination proofs, their complexity is at least EXPSPACE. We plan to refine the procedures further to achieve optimal EXPTIME complexity, e.g, by extending our deep nested sequent calculi with "global caching" techniques from tableaux systems [14].

Acknowledgment The authors would like to thank Paola Bruscoli and anonymous referrees of previous drafts of this paper for their detailed and useful comments. The first author is supported by the Australian Research Council Discovery Grant DP110103173.

Appendix

A Proofs

Lemma 4.5. The rule r is height-preserving admissible in DKm(S).

Proof. Suppose Π is a derivation of $\Gamma, (a)\{\Delta\}$. We show by induction on $|\Pi|$ that there exists a derivation Π' of $(\bar{a})\{\Gamma\}, \Delta$ such that $|\Pi| = |\Pi'|$. This is mostly straightforward, except for the case where Π ends with a propagation rule. In this case, it is enough to show that the propagation automata for $\Gamma, (a)\{\Delta\}$ is in fact exactly the same as the propagation automata of $(\bar{a})\{\Gamma\}, \Delta$. □

Lemma 4.7. The rules *actr* and m are height-preserving admissible.

Proof. Admissibility of *actr* is trivial. To show admissibility of m, the non-trivial case is when we need to permute m over p_S. Suppose Π is a derivation of $\Gamma[(a)\{\Delta_1\}, (a)\{\Delta_2\}]$ ending with a propagation rule. Suppose i is the node where Δ_1 is located and j is the node where Δ_2 is located. If $\mathcal{P}$ is a propagation automata between nodes k and l in $\Gamma[(a)\{\Delta_1\}, (a)\{\Delta_2\}]$, then $\mathcal{P}[j := i]$ is a propagation automata between nodes $k[j := i]$ and $l[j := i]$ in $\Gamma[(a)\{\Delta_1, \Delta_2\}]$. So all potential propagations of diamond formulae are preserved in the conclusion of m. So m can be permuted up over the propagation rule and by the induction hypothesis it can be eventually eliminated. □

Lemma 4.9. The structural rules $\rho(S)$ of SKm(S) are height-preserving admissible in DKm(S).

Proof. Suppose Π is a derivation of $\Gamma[(a)\{\Delta\}]$. We show that there is a derivation Π' of $\Gamma[(u)\{\Delta\}]$, where $u = a_1 \cdots a_n$ such that $a \to u \in S$. This is mostly straightforward except when Π ends with a propagation rule. Suppose the hole in $\Gamma[\ ]$ is located at node k and Δ is located at node l, with $k \succ^a l$. In this case we need to show that if a diamond formula $\langle b\rangle A$ can be propagated from a node i to node j in $\Gamma[(a)\{\Delta\}]$ then there is also a propagation path between i and j in $\Gamma[(u)\{\Delta\}]$ for the same formula. Suppose $\mathcal{P}_1$ is the propagation automata $\mathcal{R}(\Gamma[(a)\{\Delta\}], i, j)$. Then the propagation automata $\mathcal{P}_2 = \mathcal{R}(\Gamma[(u)\{\Delta\}], i, j)$ is obtained from $\mathcal{P}_1$ by adding $n-1$ new states $k_1, \dots, k_{n-1}$ between k and l, and the following transitions: $k \xrightarrow{a_1} k_1$, $k_1 \xrightarrow{a_{m+1}} k_{m+1}$, for $2 \le m < n$ and $k_{n-1} \xrightarrow{a_n} l$, and their dual transitions.

Suppose $i \xrightarrow{v} j$ is a propagation path in $\Gamma[(a)\{\Delta\}]$. If v does not go through the edge $k \succ^a l$ (in either direction, up or down) then the same path also exists in $\Gamma[(u)\{\Delta\}]$. If it does pass through $k \succ^a l$, then the path must contain one or more transitions of the form $k \xrightarrow{a} l$ or $l \xrightarrow{\bar{a}} k$. Then one can simulate the path $i \xrightarrow{v} j$ with a path $i \xrightarrow{v'} j$ in $\mathcal{P}_2$, where v' is obtained from v by replacing each $k \xrightarrow{a} l$ with $k \xrightarrow{u} l$ and each $l \xrightarrow{\bar{a}} k$ with $l \xrightarrow{\bar{u}} k$. It remains to show that $v' \in \mathcal{P}_2 \cap L_b(S)$. But this follows from the fact that $a \to u \in S$ and $\bar{a} \to \bar{u} \in S$ (because S is a closed), so $v \Rightarrow_S v' \in L_b(S)$. □

Theorem 4.10. For every context-free closed semi-Thue system S, the proof systems SKm(S) and DKm(S) are equivalent.

Proof. One direction, from SKm(S) to DKm(S) follows from the admissibility of structural rules of SKm(S) in DKm(S). To show the other direction, given a derivation Π in DKm(S), we show, by induction on the number of occurrences of p_S, with a subinduction on the height of Π, that Π can be transformed into a derivation in SKm(S). As rules other than p_S can be derived directly in SKm(S), the only interesting case to consider is when Π ends with p_S:

$$\frac{\Gamma[\langle a\rangle A]_i[A]_j}{\Gamma[\langle a\rangle A]_i[\emptyset]_j}\ p_S,\ \text{where } \mathcal{R}(\Gamma[\ ]_i[\ ]_j, i, j) \cap L_a(S) \neq \emptyset$$

and $\Gamma[\langle a\rangle A]_i[A]_j$ is derivable via a derivation Π' in DKm(S). Choose some $u \in \mathcal{R}(\Gamma[\ ]_i[\ ]_j, i, j) \cap L_a(S)$. Then we can derive the implication $\langle u\rangle A \supset \langle a\rangle A$ in SKm(S). Using this implication, the display property and the cut rule, it can be shown that the following rule is derivable in SKm(S).

$$\frac{\Gamma[\langle a\rangle A, \langle u\rangle A]}{\Gamma[\langle a\rangle A]}\ d$$

Then it is not difficult to show that the rule p_S can be simulated by the derived rule d above, with chains of r and $\langle a\rangle$-rules in SKm(S), and utilising the weakening lemma (Lemma 4.4). □

Theorem 5.2. Let S be a regular closed semi-Thue system over Σ and let $\mathcal{A}$ be a FSA representing the regular language generated by S and Σ. Then DKm(S) and DKm($\mathcal{A}$) are equivalent.

Proof. *(Outline).* In one direction, i.e., showing that a derivation of a formula B in DKm(S) can be translated into a derivation, we do induction on the height of derivations in DKm(S). As all non-propagation rules between the two systems are identical, it is enough to show that the propagation rules p_S of DKm(S) is admissible in DKm($\mathcal{A}$), which we show next.

Suppose we have a derivation Π of $\Gamma[\langle a\rangle A]_i[A]_j$ in DKm($\mathcal{A}$), and suppose that $\mathcal{R}(\Gamma[]_i[]_j, i, j) \cap L_a(S) \neq \emptyset$. We show that $\Gamma[\langle a\rangle A]_i[\emptyset]_j$ is derivable in DKm($\mathcal{A}$). In other words, this says that p_S is admissible in DKm($\mathcal{A}$).

Since $\mathcal{R}(\Gamma[]_i[]_j, i, j) \cap L_a(S) \neq \emptyset$, there must exist a sequence of transitions in $\mathcal{R}(\Gamma[\ ]_i[\ ]_j, i, j)$: $i \xrightarrow{a_1} i_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} i_{n-1} \xrightarrow{a_n} j$, where $a_1 \cdots a_n \in L_a(S)$ and where each i_k, for $1 \leq k \leq n-1$, is a node in $\Gamma[\ ]_i[\ ]_j$ and

- either $i \succ^{a_1} i_1$ or $i_1 \succ^{\bar{a}_1} i$,
- either $i_{k-1} \succ^{a_k} i_k$ or $i_k \succ^{\bar{a}_k} i_{k-1}$, for $2 \leq k < n-1$
- and either $i_{n-1} \succ^{a_n} j$ or $j \succ^{\bar{a}_n} i_{n-1}$.

Since $\mathcal{A}$ accepts $L_a(S)$, there must exist a sequence of transitions in $\mathcal{A}$ such that: $init_a \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} s_{n-1} \xrightarrow{a_n} f$, where f is a final state in $\mathcal{A}$. The propagation path $a_1 \cdots a_n$ can then be simulated in DKm($\mathcal{A}$) as follows. First, define a sequence of nested sequents as follows:

- $\Gamma_0 := \Gamma[\langle a\rangle A]_i[\emptyset]_j$, $\Gamma_1 := \Gamma[\langle a\rangle A, init_a : A]_i[\emptyset]_j$.
- $\Gamma_{k+1} := \Gamma_k(i_k \ll \{s_k : A\})$, for $1 \leq k \leq n-1$.
- $\Gamma_{n+1} := \Gamma_n(j \ll \{f : A\})$ and $\Gamma_{n+2} := \Gamma_{n+1}(j \ll \{A\})$.

Then Γ_0 can be obtained from Γ_{n+2} by a series of applications of propagation rules of DKm($\mathcal{A}$). That is, Γ_0 is obtained from Γ_1 by applying the rule i; Γ_k is obtained from Γ_{k+1} by applying either the rule $t\downarrow$ or $t\uparrow$, for $1 \leq k \leq n-1$, at node i_k and Γ_n is obtained from Γ_{n+1} by applying the rule $t\downarrow$ or $t\uparrow$ at node j, and Γ_{n+1} is obtained from Γ_{n+2} by applying the rule f at node j. Note that Γ_{n+2} is a weakening of $\Gamma[\langle a\rangle A]_i[A]_j$ with labeled formulae spread in some nodes between i and j. It remains to show that Γ_{n+2} is derivable. This is obtained simply by applying weakening to Π (this weakening lemma for DKm($\mathcal{A}$) can be proved similarly as in the proof of Lemma 4.4).

For the other direction, assume we have a DKm($\mathcal{A}$)-derivation Ψ of B. We show how to construct a derivation Ψ' of B in DKm(S). The derivation Ψ' is constructed as follows: First, remove all labelled formulae from Ψ; then remove the rules $t\uparrow$, $t\downarrow$ and i, and finally, replace the rule f with p_S. The rules $t\uparrow$, $t\downarrow$ and i from Ψ simply disappear in Ψ' because with labelled formulae removed, the premise and the conclusion of any of the rules in Ψ map to the same sequent in Ψ'. Instances of the other rules in Ψ map to the same rules in Ψ'. We need to show that Ψ' is indeed a derivation in DKm(S). The only non-trivial case is to show that the mapping from the rule f to the rule p_S is correct, i.e., the resulting instances of p_S in Ψ' are indeed valid instances. The proof is rather involved, but essentially it shows that if a labelled formula $s : A$ is present in a node in a nested sequent Γ constructed during proof search of B, then it must be a product of a sequence of propagation of labelled formulae starting from some $\langle a\rangle A$ in a node in Γ. The complete proof is omitted here, but it is available in the extended version of this paper. $\Box$

Theorem 6.4. If $Prove_1(\mathcal{A}, \{F\}) = \top$ then F is provable in DKm($\mathcal{A}$). If $Prove_1(\mathcal{A}, \{F\}) = \bot$ then F is not provable in DKm($\mathcal{A}$).

Proof. The proof of the first statement is straightforward, since the steps of $Prove_1$ are just backward applications of rules of DKm($\mathcal{A}$). To prove the second statement, we show that if $Prove_1(\mathcal{A}, \{F\}) = \bot$ then there exists a model $\mathfrak{M} = (W, R, V)$, where $R = \{R_a\}_{a\in\Sigma}$, such that $\mathfrak{M} \not\models F$. By the completeness of DKm($\mathcal{A}$), it will follow that F is not provable in DKm($\mathcal{A}$).

Since $Prove_1(\mathcal{A}, \{F\}) = \bot$ the procedure must generate an $\mathcal{A}$-stable Δ, with F in the root node of Δ. Let W be the set of all the realised nodes of Δ. For every pair $i, j \in W$, construct an automaton $\mathcal{P}(i, j)$ by modifying the propagation automaton $\mathcal{R}(\Delta, i, j)$ by identifying every unrealised node k' with its closest ancestor k such that $\Delta|k = \Delta|k'$. That is, replace every transition of the form $s \xrightarrow{a} k'$ with $s \xrightarrow{a} k$ and $k' \xrightarrow{a} s$ with $k \xrightarrow{a} s$. Then define $R_a(x, y)$ iff $\mathcal{P}(x, y) \cap L(\mathcal{A}_a) \neq \emptyset$, where $\mathcal{A}_a$ is $\mathcal{A}$ with only $init_a$ as the initial state. Suppose S is a closed semi-Thue system that corresponds to $\mathcal{A}$. Then it can be shown that the frame $\langle W, R\rangle$ satisfies S.

To complete the model, let $x \in V(p)$ iff $\neg p \in \Delta|x$. We claim that for every $x \in W$ and every $A \in \Delta|x$, we have $\mathfrak{M}, x \not\models A$. We shall prove this by induction on the size of A. Note that we ignore the labelled formulae in Δ; they are just a bookeeping mechanism. As F is in the root node of Δ, this will also prove $\mathfrak{M} \not\models F$. We show here the interesting case involving the diamond operators.

Suppose $\langle a\rangle A \in \Delta|x$. Assume for a contradiction that $\mathfrak{M}, x \models \langle a\rangle A$. That is, $R_a(x, y)$ and $\mathfrak{M}, y \models A$. If $R_a(x, y)$ then there is a accepting path $p_a(x, y)$ in $\mathcal{P}(x, y)$ of the form: $x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \cdots x_{n-1} \xrightarrow{a_n} x_n$, where $x_0 = x$ and $x_n = y$ such that $u = a_1 \dots a_n \in L(\mathcal{A}_a)$. Then because $u \in L(\mathcal{A}_a)$, there must be a sequence of states $s_0, s_1, \dots, s_n$ of $\mathcal{A}$ such that $s_0 = init_a \in I$ and $s_n \in F$ and the transitions between states: $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \cdots s_{n-1} \xrightarrow{a_n} s_n$. We show by induction on the length of transtions that that $s_i : A \in \Delta|x_i$ for $0 \leq i \leq n$. In the base case, because $\langle a\rangle A \in \Delta|x$, by $\mathcal{A}$-propagation, we have $s_0 : A \in \Delta|x_0$. For the inductive cases, suppose $s_i : A \in \Delta|x_i$, for $n > i \geq 0$. There are two cases to consider. Suppose the transition $x_i \xrightarrow{a_{i+1}}_{\mathcal{P}(x,y)} x_{i+1}$ is present in $\mathcal{R}(\Delta, x, y)$. Then either $x_i \succ^{a_{i+1}} x_{i+1}$ or $x_{i+1} \succ^{\bar{a}_{i+1}} x_i$. In either case, by $\mathcal{A}$-propagation of Δ, we must have $s_{i+1} : A \in \Delta|x_{i+1}$.

If $x_i \xrightarrow{a_{i+1}}_{\mathcal{P}(x,y)} x_{i+1}$ is not a transition in $\mathcal{R}(\Delta, x, y)$, then this transition must have resulted from a use of a loop node. There are two subcases: either x_i or x_{i+1} is the closest ancestor of a loop node x' with $\Delta|x_i = \Delta|x'$ or, respectively, $\Delta|x_{i+1} = \Delta|x'$. Suppose x_i is the closest ancestor of x' with $\Delta|x_i = \Delta|x'$. By the definition of $\mathcal{P}(x, y)$, this means we have $x' \xrightarrow{a_{i+1}} x_{i+1}$ in $\mathcal{R}(\Delta, x, y)$. Because $\Delta|x_i = \Delta|x'$ and $s_i : A \in \Delta|x_i$, we have $s_i : A \in \Delta|x'$. Then by $\mathcal{A}$-propagation, it must be the case that $s_{i+1} : A \in \Delta|x_{i+1}$. Suppose x_{i+1} is the closest ancestor of x' with $\Delta|x' = \Delta|x_{i+1}$. Then $x_i \xrightarrow{a_{i+1}} x'$ is a transition in $\mathcal{R}(\Delta, x, y)$. By $\mathcal{A}$-propagation, it must be the case that $s_{i+1} : A \in \Delta|x'$, and therefore also $s_{i+1} : A \in \Delta|x_{i+1}$.

So we have $s_n : A \in \Delta|y$. But, again by $\mathcal{A}$-propagation, this means $A \in \Delta|y$ (because s_n is a final state). Then by the induction hypothesis, we have $\mathfrak{M}, y \not\models A$, contradicting the assumption. $\square$

Theorem 6.5. For every formula A, $Prove_1(\mathcal{A}, \{A\})$ terminates.

Proof. *(Outline)* We say that a nested sequent Γ is a *set-based nested sequent* if in every node of Γ, every (labelled) formula occurs at most once (a formula C and its labelled versions are considered distinct). By inspection of the procedure $Prove_1$, it is clear that all the intermediate sequents created during proof search for $Prove_1(\mathcal{A}, \{A\})$ are set-based sequents.

The only possible cause of non-termination is steps (v) and (vi), where the input nested sequent is extended with new nodes. The blocking condition in (vi) ensures that the height of any nested sequent generated during proof search is bounded. Let m be the number of states in $\mathcal{A}$ and let n be the number of subformulae of A. Then the total number of different sets of formulae and labeled formulae (with labels from $\mathcal{A}$) is bounded by $2^{(m+1)n}$. Therefore, any set-based nested sequent generated during proof search will have height at most $2^{(m+1)n}$, as the loop checking ensures they are never expanded beyond this height. Given a nested sequent of a fixed height, the expansion in step (v) only adds to the width of the nested sequent. This expansion is limited by the number of 'boxed' subformulae of A. So the size of the nested sequents generated during proof search is bounded, and therefore each branch of the search has a finite length, thus the procedure must terminate. $\square$

Lemma 6.9. Let S be a context-free closed semi-Thue system. If Γ is an S-stable nested sequent, then there exists a model $\mathfrak{M}$ such that for every node x in Γ there exists a world w in $\mathfrak{M}$ such that for every $A \in \Gamma|x$, we have $\mathfrak{M}, w \not\models A$.

Proof. Let $\boldsymbol{x} = x_1, \dots, x_n$ be the list of (pairwise distinct) unrealised leaf nodes in Γ. Because Γ is S-stable, we have a function λ assigning each unrealised leaf node x_i to an ancestor node $\lambda(x_i)$ such that $\Gamma|x_i = \Gamma|\lambda(x_i)$, and for every node y and z in Γ, we have that Γ is $(S, \mathcal{P}(y,z))$-propagated, where $\mathcal{P}(y,z) = \mathcal{R}(\Gamma, y, z)[\boldsymbol{x} := \lambda(\boldsymbol{x})]$. Then define $\mathfrak{M} = \langle W, \{R_a \mid a \in \Sigma\}, V\rangle$ where

- W is the set of nodes of Γ minus the nodes $\boldsymbol{x}$,
- for every $x, y \in W$, $R_a(x,y)$ iff $\mathcal{P}(x,y) \cap L_a(S) \neq \emptyset$, and
- $V(p) = \{x \in W \mid \neg p \in \Gamma|x\}$.

We now show that for every node v in Γ, there exists a $w \in W$ such that if $A \in \Gamma|v$ then $\mathfrak{M}, w \not\models A$, where the world w is determined by v as follows: if v is in $\boldsymbol{x}$, then $w = \lambda(v)$; otherwise, $w = v$. We prove this by induction on the size of A. The only interesting cases are those where $A = \langle a\rangle C$ or $A = [a]C$ for some a and C.

- Suppose $A = \langle a \rangle C$. Suppose, for a contradiction, that $\mathfrak{M}, w \models \langle a \rangle C$. That means there exists a w' such that $R_a(w, w')$ and $\mathfrak{M}, w' \models C$. By the definition of R_a, we have that $\mathcal{P}(w, w') \cap L_a(S) \neq \emptyset$. Because Γ is S-stable, by Definition 6.8(iv), it is $(S, \mathcal{P}(w, w'))$-propagated. This means that $C \in \Gamma|w'$. Then by the induction hypothesis, $\mathfrak{M}, w' \not\models C$, which contradicts our assumption.
- Suppose $A = [a]C$. To show $\mathfrak{M}, w \not\models [a]C$, it is enough to show there exists w' such that $R_a(w, w')$ and $\mathfrak{M}, w' \not\models C$.

  Note that w must be an internal node in Γ, so by the S-stability of Γ, node w in Γ must be realised. Therefore there exists a node z such that $w \succ^a z$ in Γ and $C \in \Gamma|z$. If $z \notin \boldsymbol{x}$, then let $w' = z$; otherwise, let $w' = \lambda(z)$. In either case, $\Gamma|z = \Gamma|w'$, so in particular, $C \in \Gamma|w'$. Also, in either case, the propagation automata $\mathcal{P}(w, w')$ contains a transition $w \xrightarrow{a}_{\mathcal{P}(w,w')} w'$ (in the case where $z \in \boldsymbol{x}$, this is because $\lambda(z)$ is identified with z). Obviously, $a \in L_a(S)$, so $L_a(S) \cap \mathcal{P}(w, w') \neq \emptyset$, so by the definition of R_a, we have $R_a(w, w')$. Since $C \in \Gamma|w'$, by the induction hypothesis, $\mathfrak{M}, w' \not\models C$. So we have $R_a(w, w')$, and $\mathfrak{M}, w' \not\models C$, therefore $\mathfrak{M}, w \not\models [a]C$.

□

Theorem 6.10. Let S be a context-free closed semi-Thue system. For every formula F, $\mathit{Prove}(S, \{F\})$ returns $\top$ if and only if F is provable in DKm(S).

Proof. *(Outline)* One direction, i.e., $\mathit{Prove}(S, \{F\}) = \top$ implies that F is provable in DKm(S), follows from the fact that steps of *Prove* are simply backward applications of rules of DKm(S). To prove the other direction, we note that if F has a derivation in DKm(S), it has a derivation of a minimal length, say Π. In particular, in such an derivation, there are no two identical nested sequents in any branch of the derivation. Because in DKm(S) each backward application of a rule retains the principal formula of the rule, every application of a rule in Π will eventually be covered by one of the steps of *Prove*. Since there are only finitely many rule applications in Π, eventually these will all be covered by *Prove* and therefore it will terminate. For example, if Π ends with a diamond (propagation) rule applied to a non-saturated sequent, the *Prove* procedure will choose to first saturate the sequent before applying the propagation rule. Since all rules are invertible, we do not lose any provability of the original sequent, but the *Prove* procedure may end up doing more steps. We need to show, additionally, that every sequent arising from the execution of $\mathit{Prove}(S, \{F\})$ is not S-stable. Suppose otherwise, i.e., the procedure produces an S-stable sequent Δ. Now it must be the case that F is in the root node of Δ. By Lemma 6.9, this means there exists a countermodel that falsifies F, contrary to the validity of F. □

Theorem 6.11. Let S be a regular closed semi-Thue system. Then for every formula F, the procedure $\mathit{Prove}(S, \{F\})$ terminates.

Proof. Since S is regular, there exists a minimal deterministic FSA $\mathcal{A}$ corresponding to S such that $\mathit{Prove}_1(\mathcal{A}, \{F\})$ terminates.

Suppose $\mathit{Prove}_1(\mathcal{A}, \{F\}) = \top$. Then F must be derivable in DKm($\mathcal{A}$)

by Theorem 6.4. Since DKm($\mathcal{A}$) and DKm(S) are equivalent (Theorem 5.2), there must also be a derivation of F in DKm(S). Then by Theorem 6.10, $Prove(S, \{F\})$ must terminate and return $\top$.

Suppose $Prove_1(\mathcal{A}, \Gamma) = \bot$. Then there exists an $\mathcal{A}$-stable Γ' that can be constructed from Γ in the execution of $Prove_1(\mathcal{A}, \Gamma)$. It can be shown that a Δ that is identical to Γ' without any labelled formulae can be constructed in the execution of $Prove_2(S, \Gamma, d)$ for some d. We claim that Δ is S-stable. Saturation, propagation and the realisation of internal nodes follow immediately from the construction, it remains to find a function λ as in Definition 6.8. We claim that such a function is given by $\lambda(x) = y$ where y is the closest ancestor of x in Γ' such that $\Gamma'|x = \Gamma'|y$. That is, we identify each unrealised leaf with the same node it would have been identified with in $Prove_1(\mathcal{A}, \Gamma)$.

Let $\boldsymbol{i} = i_1, \ldots, i_l$ be the list of all unrealised leaf nodes in Δ and let $\mathcal{P}(x, y) = \mathcal{R}(\Delta, x, y)[\boldsymbol{i} := \lambda(\boldsymbol{i})]$. (Note that as the tree structures of Γ' and Δ are identical, we also have $\mathcal{P}(x, y) = \mathcal{R}(\Gamma', x, y)[\boldsymbol{i} := \lambda(\boldsymbol{i})]$.) For a contradiction, suppose there exists j and k such that Δ is not $(S, \mathcal{P}(j, k))$-propagated, i.e., there exist $\langle a \rangle A \in \Delta|j$, such that $A \notin \Delta|k$ but $\mathcal{P}(j, k) \cap L_a(S) \neq \emptyset$. In other words, there is a word $b_1 \ldots b_n \in \mathcal{P}(j, k) \cap L_a(S)$, and a sequence of states $x_0, \ldots, x_n$ in $\mathcal{P}(j, k)$ such that $x_0 = j, x_n = k, x_{m-1} \xrightarrow{b_m}_{\mathcal{P}(j,k)} x_m$, where $1 \leq m < n$. We will show that there exists a function St assigning states of $\mathcal{A}$ to nodes of Γ' satisfying: $St(x_0) \in I$, $St(x_{m-1}) \xrightarrow{b_m}_{\mathcal{A}} St(x_m)$, $St(x_n) \in F$, and $St(x_m) : A \in \Gamma'|x_m$. This will establish that $St(x_n) : A \in \Gamma'|x_n$ where $St(x_n) \in F$. Then by $\mathcal{A}$-propagation, it will follow that $A \in \Gamma'|k$, and therefore $A \in \Delta|k$, contradicting our assumption that $A \notin \Delta|k$.

Let $s_0, \ldots, s_n$ be the run of $\mathcal{A}_a$ associated with input $b_1 \ldots b_n$. Let $St(x_m) = s_m$. As $L(\mathcal{A}_a) = L_a(S)$, we know that $s_0, \ldots, s_n$ is an accepting run. This gives us $St(x_0) \in I, St(x_{m-1}) \xrightarrow{b_m}_{\mathcal{A}} St(x_m)$ and $St(x_n) \in F$. It remains to show that $St(x_m) : A \in \Gamma'|x_m$. We will do so by induction on m.

Base case: As $\langle a \rangle A \in \Gamma'|x_0$, by $\mathcal{A}$-propagation we obtain $s_0 : A \in \Gamma'|x_0$.

Inductive case: Suppose $x_m \xrightarrow{b_{m+1}}_{\mathcal{P}(j,k)} x_{m+1}$. By the inductive hypothesis, $s_m : A \in \Gamma'|x_m$. There are two cases to consider:

- The transition $x_m \xrightarrow{b_{m+1}}_{\mathcal{P}(j,k)} x_{m+1}$ also exists in $\mathcal{R}(\Gamma', j, k)$. In this case, by $\mathcal{A}$-propagation, we have $s_{m+1} : A \in \Gamma'|x_{m+1}$.
- The transition $x_m \xrightarrow{b_{m+1}}_{\mathcal{P}(j,k)} x_{m+1}$ is obtained from $\mathcal{R}(\Gamma', j, k)$ through the identification of unrealised leaf nodes with their closest ancestors. There are two subcases:
  - · $x_m = \lambda(y)$ for some unrealised leaf node y such that $\Gamma'|x_m = \Gamma'|y$, and $y \xrightarrow{b_{m+1}}_{\mathcal{R}(\Gamma',j,k)} x_{m+1}$. Since $\Gamma'|x_m = \Gamma'|y$, we have that $s_m : A \in \Gamma'|y$ and it follows by $\mathcal{A}$-propagation that $s_{m+1} : A \in \Gamma'|x_{m+1}$.
  - · $x_{m+1} = \lambda(y)$ for some unrealised leaf node y such that that $\Gamma'|x_{m+1} = \Gamma'|y$, and $x_m \xrightarrow{b_{m+1}}_{\mathcal{R}(\Gamma',j,k)} y$. By $\mathcal{A}$-propagation, $s_{m+1} : A \in \Gamma'|y = \Gamma'|x_{m+1}$.

Thus when $Prove(S, \Gamma)$ calls $Prove_2(S, \Gamma, d)$, it will construct an S-stable sequent and terminate. □

References

[1] Baldoni, M., L. Giordano and A. Martelli, *A tableau for multimodal logics and some (un)decidability results*, in: *TABLEAUX*, LNCS **1397** (1998), pp. 44–59.

[2] Belnap, N., *Display logic*, Journal of Philosophical Logic **11** (1982), pp. 375–417.

[3] Blackburn, P., J. van Benthem and F. Wolter, "Handbook of Modal Logic," Studies in Logic and Practical Reasoning, Elsevier, 2007.

[4] Brünnler, K., *Deep sequent systems for modal logic*, Archive for Mathematical Logic **48** (2009), pp. 551–577.

[5] Brünnler, K. and A. Tiu, *A local system for classical logic*, in: *LPAR*, LNCS **2250** (2001), pp. 347–361.

[6] Demri, S., *The complexity of regularity in grammar logics and related modal logics*, J. Log. Comput. **11** (2001), pp. 933–960.

[7] Demri, S. and H. de Nivelle, *Deciding regular grammar logics with converse through first-order logic*, Journal of Logic, Language and Information **14** (2005), pp. 289–329.

[8] Fariñas del Cerro, L. and M. Penttonen, *Grammar logics*, Logique et Analyse **121-122** (1998), pp. 123–134.

[9] Ginsburg, S., "The Mathematical Theory of Context-Free Languages," McGraw-Hill, Inc., New York, NY, USA, 1966.

[10] Goré, R. and L. A. Nguyen, *A tableau calculus with automaton-labelled formulae for regular grammar logics*, in: *TABLEAUX*, LNCS **3702** (2005), pp. 138–152.

[11] Goré, R., L. Postniece and A. Tiu, *Taming displayed tense logics using nested sequents with deep inference*, in: *TABLEAUX*, LNCS **5607** (2009), pp. 189–204.

[12] Goré, R., L. Postniece and A. Tiu, *Cut-elimination and proof search for bi-intuitionistic tense logic*, in: *Advances in Modal Logic* (2010), pp. 156–177.

[13] Goré, R., L. Postniece and A. Tiu, *On the correspondence between display postulates and deep inference in nested sequent calculi for tense logics*, Logical Methods in Computer Science **7** (2011).

[14] Goré, R. and F. Widmann, *Sound global state caching for $\mathcal{ALC}$ with inverse roles*, in: *TABLEAUX*, Lecture Notes in Computer Science **5607**, 2009, pp. 205–219.

[15] Goré, R. and F. Widmann, *Optimal and cut-free tableaux for propositional dynamic logic with converse*, in: *IJCAR*, LNCS **6173** (2010), pp. 225–239.

[16] Guglielmi, A., *A system of interaction and structure*, ACM Trans. Comput. Log. **8** (2007).

[17] Horrocks, I., O. Kutz and U. Sattler, *The even more irresistible $\mathcal{SROIQ}$*, in: *KR* (2006), pp. 57–67.

[18] Horrocks, I. and U. Sattler, *Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms*, Artif. Intell. **160** (2004), pp. 79–104.

[19] Kashima, R., *Cut-free sequent calculi for some tense logics*, Studia Logica **53** (1994), pp. 119–135.

[20] Kazakov, Y., *$\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$*, in: *KR* (2008), pp. 274–284.

[21] Kracht, M., *Power and weakness of the modal display calculus*, in: *Proof theory of modal logic (Hamburg, 1993)*, Applied Logic Series **2**, Kluwer Acad. Publ., 1996 pp. 93–121.

[22] Nguyen, L. A. and A. Szałas, *A tableau calculus for regular grammar logics with converse*, in: *CADE*, LNCS **5663**, 2009, pp. 421–436.

[23] Nguyen, L. A. and A. Szałas, *Exptime tableau decision procedures for regular grammar logics with converse*, Studia Logica **98** (2011), pp. 387–428.

[24] Ramanayake, D. R. S., "Cut-elimination for provability logics and some results in display logic," Ph.D. thesis, The Australian National University (2011).

[25] Simpson, A. K., "The Proof Theory and Semantics of Intuitionistic Modal Logics," Ph.D. thesis, University of Edinburgh (1994).