Optimizing Coalgebraic Modal Logic Reasoning

Daniel Hausmann*

Department of Computer Science, Universität Bremen
hausmann@informatik.uni-bremen.de

Abstract. The framework provided by coalgebraic modal logics offers broadly applicable coalgebraic semantics and an ensuing general treatment of modal sequent and tableau calculi while covering a wide variety of logics ranging from graded and probabilistic modal logic to coalition logic and conditional logics. Here we discuss generic optimisation strategies that may be employed to improve the performance of global caching algorithms that decide the satisfiability of coalgebraic modal logics. Specifically, we discuss and show the admissability of generalisations of such established strategies as propositional and modal *simplification*, *dependency directed backtracking*, *semantic branching* and complex *heuristics* for coalgebraic modal logics. As a more advanced consideration, the flattened representation of the involved proof graph by a *proof formula* is shown to be sound and complete; this separation of *proof structure* from the actual content of the proof does not only enhance the performance of the propagation process, it also allows for further optimisation techniques such as *proof graph simplification* to be applied. The relevance of the theoretic results is exhibited by detailled benchmarking of the examplary implementation, the *Coalgebraic Logic Satisfiability Solver* (*CoLoSS*), against established testsets of formulas.

1 Introduction

...

2 Coalgebraic Sequent Calculus

2.1 Notation

Given an endofunctor on sets T, a T-coalgebra $\mathcal{C}$ consists of a carrier-set C and a function $\gamma : C \to T(C)$.

The semantics of modal operators $\heartsuit \in \Lambda$ is then defined by means of *predicate liftings* $[\![\heartsuit]\!] : 2^n \to 2 \circ T^{op}$ where $2 : \mathbf{Set} \to \mathbf{Set}^{op}$ denotes the contravariant powerset functor.

Formulas are of the shape

* Work forms part of DFG-project *Generic Algorithms and Complexity Bounds in Coalgebraic Modal Logic* (SCHR 1118/5-1)

$$\mathcal{F}(\Lambda) \ni A_1, \ldots, A_n := p \mid \neg A_1 \mid A_1 \wedge A_2 \mid A_1 \vee A_2 \mid \heartsuit(A_1, \ldots, A_n)$$

where $\heartsuit \in \Lambda$ is an n-ary modal operator.

The semantics with respect to a model coalgebra $M = (C, \gamma, \pi)$ (where π denotes a valuation of propositional atoms) is fixed as follows:

$$[\![\heartsuit(A_1, \ldots, A_n)]\!]_M = \gamma^{-1} \circ [\![\heartsuit]\!]_C([\![A_1]\!]_M, \ldots, [\![A_n]\!]_M)$$

A (Λ-)*sequent* Γ is a set of formulas (from $\mathcal{F}(\Lambda)$).

A *sequent rule* consists of its *premise* Σ (a set of sequents) and its *conclusion* Γ (one sequent). A rule $R = (\Sigma = \{\Gamma_1, \ldots, \Gamma_n\}, \Gamma)$ is usually presented as follows:

$$\frac{\Gamma_1 \qquad \ldots \qquad \Gamma_n}{\Gamma}$$

A rule application of rule R to sequent Γ is an instantiation of R s.t. the conclusion of the rule is equal to Γ.

A formula ϕ is provable if there is a model coalgebra $M = (C, \gamma, \pi)$ with a state $x \in C$ s.t. $[\![\phi]\!] \ni x$. A sequent $\Gamma = \{\phi_1, \ldots, \phi_n\}$ is provable iff $\bigvee_{i=1}^{n} \phi_i$ is provable. A premise is said to be provable iff all its sequents are provable.

2.2 A generic sequent calculus for coalgebraic modal logics

A sound and complete sequent calculus (w.r.t. the according coalgebraic logic) is obtained if the propositional rules from Figures 1 and 2 are utilized together with the appropriate modal rule(s). Some examplary modal rules for different coalgebraic modal logics are depicted in Figure 3 while the modal rules of some conditional logics are introduced in Figure 4.

$$\boxed{(\mathrm{T})\frac{}{\Gamma, \top} \qquad (\mathrm{At})\frac{}{\Gamma, p, \neg p} \qquad (\neg\vee)\frac{\Gamma, A \qquad \Gamma, B}{\Gamma, \neg(\neg A \vee \neg B)}}$$

Fig. 1. Branching or finishing propositional sequent rules

$$\boxed{(\neg\neg)\frac{\Gamma, A}{\Gamma, \neg\neg A} \qquad (\vee)\frac{\Gamma, A, B}{\Gamma, (A \vee B)}}$$

Fig. 2. Linear propositional sequent rules

Take note that the rules from Figure 2 may be integrated into the normalisation process (see 3.1) due to their linearity.

Modal Logic (Feature)	Modal Rule(s)
M ($\Box_{\mathbf{M}}$)	$\dfrac{A \to B}{\Gamma, \Box_{\mathbf{M}} A \to \Box_{\mathbf{M}} B}$
K ($\Box_{\mathbf{K}}$)	$\dfrac{\bigwedge_{i=1}^{n} A_i \to B}{\Gamma, \bigwedge_{i=1}^{n} \Box_{\mathbf{K}} A_i \to \Box_{\mathbf{K}} B}$
KD ($\Box_{\mathbf{KD}}$)	$\dfrac{\bigwedge_{i=1}^{n} A_i \to B}{\Gamma, \bigwedge_{i=1}^{n} \Box_{\mathbf{KD}} A_i \to \Box_{\mathbf{KD}} B}$ $\qquad$ $\dfrac{\neg \bigwedge_{i=1}^{n} A_i}{\Gamma, \neg \bigwedge_{i=1}^{n} \Box_{\mathbf{KD}} A_i}$
Hennessy-Milner-Logic ($\Box^{a}_{\mathbf{HM}}$)	$\dfrac{\bigwedge_{i=1}^{n} A_i \to B}{\Gamma, \bigwedge_{i=1}^{n} \Box^{a}_{\mathbf{HM}} A_i \to \Box^{a}_{\mathbf{HM}} B}$
Coalition Logic ($\Box^{C}_{\mathbf{C}}$)	$\dfrac{\bigwedge_{i=1}^{n} A_i \to B \vee \bigvee_{j=1}^{m} C_j}{\Gamma, \bigwedge_{i=1}^{n} \Box^{C_i}_{\mathbf{C}} A_i \to \Box^{D}_{\mathbf{C}} B \vee \bigvee_{j=1}^{m} \Box^{N}_{\mathbf{C}} C_j}$ $m, n \geq 0$, C_i pairwise disjoint subsets of D.
Graded Modal Logic ($\Diamond^{i}_{\mathbf{G}}$)	$\dfrac{\sum_{i=1}^{n} A_i \leq \sum_{j=1}^{m} B_j}{\Gamma, \bigwedge_{i=1}^{n} \Diamond^{k_i}_{\mathbf{G}} A_i \to \bigvee_{j=1}^{m} \Diamond^{l_j}_{\mathbf{G}} B_j}$ $n, m \geq 0$, $\sum_{i=1}^{n} (k_i + 1) \geq 1 + \sum_{j=1}^{m} l_j$.
Probabilistic Modal Logic ($\Diamond^{p}_{\mathbf{P}}$)	$\dfrac{\sum_{i=1}^{n} A_i + u \leq \sum_{j=1}^{m} B_j}{\Gamma, \bigwedge_{i=1}^{n} \Diamond^{p_i}_{\mathbf{P}} A_i \to \bigvee_{j=1}^{m} \Diamond^{q_j}_{\mathbf{P}} B_j}$ $m, n \geq 0$, $m + n \geq 1$, $u \in \mathbb{Z}$, $\sum_{i=1}^{n} p_i + u \geq \sum_{j=1}^{m} q_j$ and $m = 0 \to \sum_{i=1}^{n} p_i + u > 0$.

Fig. 3. One-step complete, resolution closed modal rules for some coalgebraic modal logics

We introduce the following abbreviations in order to allow for a swift presentation of the modal rules of propositional and graded modal logic: Given a formula ϕ_i and $r_i \in \mathbb{Z}$ for all $i \in I$ as well as $k \in \mathbb{Z}$,

$$\sum_{i\in I} r_i\phi_i \geq k \equiv \bigwedge_{J\subseteq I, r(J)<k} (\bigwedge_{j\in J}\phi_j \to \bigvee_{j\notin J}\phi_j),$$

where $r(J) = \sum_{j\in J} r_j$. Furthermore, we use $\sum a_i \leq \sum b_j$ as an abbreviation for $\sum b_j - \sum a_i \geq 0$.

Modal Logic (Feature)	Modal Rule(s)
CK ($\Rightarrow_{\mathbf{CK}}$)	$\dfrac{A_0 \leftrightarrow \ldots \leftrightarrow A_n \qquad \neg B_1, \ldots, \neg B_n, B_0}{\Gamma, \bigwedge_{i=1}^{n}(A_i \Rightarrow_{\mathbf{CK}} B_i) \to (A_0 \Rightarrow_{\mathbf{CK}} B_0)}$
CK+CEM ($\Rightarrow_{\mathbf{CK_{CEM}}}$)	$\dfrac{A_0 \leftrightarrow \ldots \leftrightarrow A_n \qquad B_0, \ldots, B_k, \neg B_{k+1}, \ldots \neg B_n}{\Gamma, \bigwedge_{i=k+1}^{n}(A_i \Rightarrow_{\mathbf{CK_{CEM}}} B_i) \to \bigvee_{j=0}^{k}(A_j \Rightarrow_{\mathbf{CK_{CEM}}} B_j)}$
CK+CM ($\Rightarrow_{\mathbf{CK_{CM}}}$)	$\dfrac{\begin{array}{ll} A_i = A_j & \text{for } i,j \in l(v), v \in G \text{ initial} \\ \bigcup_{i\in l(v)}\{\neg B_i, \neg A_i\}, A_j & \text{for } v \in G \text{ and } j \in l(w) \\ & \text{for some successor } w \text{ of } v \\ \{\neg B_i \mid i \in I\}, D & \\ \bigcup_{i\in I}\{\neg B_i, \neg A_i\}, C & \\ \neg C, A_i & \text{for } i \in I \end{array}}{\Gamma, \bigwedge_{i\in I}(A_i \Rightarrow_{\mathbf{CK_{CM}}} B_i) \to (C \Rightarrow_{\mathbf{CK_{CM}}} D)}$ for an I-compatibility graph G
System S ($\Rightarrow_{SysS}$)	$\dfrac{\Delta_M(\nu(M)) \text{ for each } M \in \mathfrak{S}_{\Gamma_0}}{\underbrace{\Gamma, \bigwedge_{i\in I}(A_i \Rightarrow_{SysS} B_i) \to (A_0 \Rightarrow_{SysS} B_0)}_{\equiv:\Gamma_0}}$ where $\mathfrak{S}_{\Gamma_0}$ denotes all S-structures $(S, \preceq)$ over Γ_0, $\Delta_M([i]) \equiv \bigcup_{k\simeq i}\{\neg A_k, \neg B_k\}, \{A_j \mid i \prec j \vee j \notin S\}$, $\Delta_M([0]) \equiv \neg A_0, \bigcup_{0\neq k\simeq 0}\{\neg A_k, \neg B_k\}, B_0, \{A_j \mid j \notin S\}$.

Fig. 4. One-step complete, resolution closed modal rules for some conditional coalgebraic logics

Given a sequent Γ, a proof tree $t = (s, p, r_1, r_2, x)$ is a two-kinded tree with sequents s and premises p as nodes and a set x of unexpanded sequents. The relation $r_1 : s \to \mathcal{P}(p)$ assigns to each sequent the set of its successor premises, whereas the relation $r_2 : p \to \mathcal{P}(s)$ returns for each premise the set of sequents out of which it consists. A proof tree has its root at the sequent node Γ.

For all expanded sequents $\Gamma \notin x$, $r_1(\Gamma) = \{\Sigma \mid (\Sigma, \Gamma)$ is an instance of a rule $R\}$. For all premises Σ, $r_2(\Sigma) = \{\Gamma \mid \Sigma \in \Gamma\}$.

Unexpanded sequents $\Gamma \in x$ (i.e. sequents to which no rule has been applied yet), are called *open sequents*. A proof tree with $x = \emptyset$ is called *fully expanded*.

Any sequent to which either the rule (T) or the rule (At) may be applied is called a *successful leaf* of the proof tree.

Any sequent to which no rule may be applied is called an *unsuccessful leaf.* A proof tree with root Γ is successful if

1. it is a successful leaf, or
2. there is *any* $\Sigma \in r_1(\Gamma)$, s.t. *all* $\Gamma' \in r_2(\Sigma)$ have a successful proof tree.

For the sake of readability, we will usually show *simplified proof trees* where only one rule application is considered for each sequent (such that only one – the interesting one – of the possible premises of each sequent is shown).

It was shown in Schröder, Pattinson, 2008, that this generic sequent calculus enables the construction of a PSPACE decision procedure for the provability of the according coalgebraic logic – as long as the used modal rules are one-step complete and resolution closed as well as... [ask Lutz for name of property]

3 Optimisations

An optimisation technique for provability proofs of coalgebraic modal formulas is a coalgebra homomorphism, mapping states of a witness coalgebra to states of an optimized witness coalgebra (given a formula ϕ, coalgebra (A, α) is a witness of ϕ, whenever there is a state $a \in A$ s.t. $\phi \in a$). In general, the idea is to map bigger coalgebras to smaller coalgebras, thusly reducing the amount of necessary proof work.

In general, we are interested in finding as small as possible proofs for a given formula. *Simplification by tautologies* allows us to cut of short successful branches or to shorten long successful branches in the proof tree (without the need to completely explore them). *Semantic branching* allows to ensure that two or more sub-proofs (which are induced by a conjunction or a modal operator) are strictly disjoint in order to avoid repeated treatment of a sepcific formula. The technique of *controlled backtracking* allows us to cut of all branches in the proof tree which contain formulas that have already been shown to lead to success. *Heuristics* try to choose the shortest path through the proof tree that leads to success. Finally, the use of *caching* techniques turns (parts of) the proof tree into a proof graph, thusly eliminating the need for duplicate proofs while paying the price of possibly exponential storage needs.

The most important expectation with regards to any optimisation is of course that it must respect the soundness and completeness of the algorithm. Furthermore any optimisation involves some computations and might hence sometimes *decrease* the efficiency of the underlaying algorithm. Thus another important property of optimisations is, that its gain should 'usually' be larger than the invested effort.

We describe the optimisation techniques (all of which being adoptations or generalisations of the techniques from Horrocks,Patel-Schneider, except the restriction to maximal applications of modal rules - Horrocks,Patel-Schneider considered only logics for which this restriction is admissable) in more detail:

3.1 Normalisation

First of all, we fix the syntactic form in which we assume all formulas to be. This is achieved by the process of *normalisation* of formulas which is in fact not an optimisation by itself, but rather a necessary condition for the usage of optimisations such as simplification by tautologies.

Syntactic construct	Normalisation
$\bot$	$\neg\top$
$\neg\neg\phi$	ϕ
$\phi \wedge \psi$	$\neg(\neg\phi \vee \neg\psi)$
$\phi \rightarrow \psi$	$\neg\phi \vee \psi$
$\phi \leftrightarrow \psi$	$(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$
$\phi \vee \psi$	$\cup\{\phi, \psi\}$
$\cup\{\phi, (\cup\Gamma)\}$	$\cup(\{\phi\} \cup \Gamma)$

Fig. 5. Normalisation rules

The input formula is assumed to already contain only the standard modalities for each feature. Thus we normalize to truth, negation and generalized disjunction (where the collection of all disjuncts is treated as a set in order to remove duplicates and ordering).

3.2 Simplification by Tautologies

Optimisation by simplifaction rules makes use of specific (propositional or modal) tautologies in order to deterministically rewrite any appearing formulas in a proof. The application of simplification does not invoke any branching and is thus a linear process.

However, only tautologies with very few (and hence quickly checked for) assumptions should be used for simplification. Checking the assumptions of tautologies may involve the checking of syntactical equivalence of formulas. This task may be optimized by computing unique hashes for formulas and by then only checking for equality of the hash-values.

It is possible to distinguish two kinds of rules for the optimisation by simplification: Generic propositional rules which are the same for any coalgebraic modal logic and more specific modal simplification rules which may differ between any two concrete modal logics:

Propositional Simplification The following is a non-exhaustive set of prositional simplification rules on sequents:

The last propositional tautology represents boolean constraint propagation (BCP).

Operator	Propositional Tautology
$\vee$	$\cup\{\top, ...\} \rightarrow \top$ $\cup\{\neg\top, \phi_1, \ldots, \phi_n\} \rightarrow \cup\{\phi_1, \ldots, \phi_n\}$ $\cup\{\phi, \neg\phi, \ldots\} \rightarrow \top$
$\vee\wedge$	$\cup\{\neg\phi, \neg(\neg\phi \vee \neg\psi)\} \rightarrow \psi$

Fig. 6. Propositional tautologies used for simplification

Modal Simplification A modal feature may or may not allow for reasonable simplification:

Feature	Modal Tautology
$\Box_{\mathbf{M}}$	$\Box_{\mathbf{M}}\top \rightarrow \top$
$\Box_{\mathbf{K}}$	$\Box_{\mathbf{K}}\top \rightarrow \top$ $(\Box_{\mathbf{K}}\phi \wedge \Box_{\mathbf{K}}\psi \rightarrow \Box_{\mathbf{K}}(\phi \wedge \psi))$
$\Box_{\mathbf{KD}}$	$\Box_{\mathbf{KD}}\top \rightarrow \top$ $(\Box_{\mathbf{KD}}\phi \wedge \Box_{\mathbf{KD}}\psi \rightarrow \Box_{\mathbf{KD}}(\phi \wedge \psi))$ $\neg\Box_{\mathbf{KD}}\bot \rightarrow \top$
$\Box^a_{\mathbf{HM}}$	$\Box^a_{\mathbf{HM}}\top \rightarrow \top$ $(\Box^a_{\mathbf{HM}}\phi \wedge \Box^a_{\mathbf{HM}}\psi \rightarrow \Box^a_{\mathbf{HM}}(\phi \wedge \psi))$
$\Box^C_{\mathbf{C}}$	$\Box^C_{\mathbf{C}}\top \rightarrow \top$
$\Diamond^i_{\mathbf{G}}$	
$\Diamond^p_{\mathbf{P}}$	
$\Rightarrow_{\mathbf{CK}}$	$(\phi \Rightarrow_{\mathbf{CK}} \top) \rightarrow \top$
$\Rightarrow_{\mathbf{CK_{CEM}}}$	$(\phi \Rightarrow_{\mathbf{CK_{CEM}}} \top) \rightarrow \top$
$\Rightarrow_{\mathbf{CK_{CM}}}$	$(\phi \Rightarrow_{\mathbf{CK_{CM}}} \top) \rightarrow \top$ $((\top \Rightarrow_{\mathbf{CK_{CM}}} \bot) \rightarrow (\phi \Rightarrow_{\mathbf{CK_{CM}}} \psi)) \rightarrow \top$ $((\top \Rightarrow_{\mathbf{CK_{CM}}} \phi) \rightarrow (\top \Rightarrow_{\mathbf{CK_{CM}}} \top)) \rightarrow \top$ $((\phi \Rightarrow_{\mathbf{CK_{CM}}} \bot) \rightarrow (\bot \Rightarrow_{\mathbf{CK_{CM}}} \psi)) \rightarrow \top$ $((\bot \Rightarrow_{\mathbf{CK_{CM}}} \phi) \rightarrow (\bot \Rightarrow_{\mathbf{CK_{CM}}} \phi)) \rightarrow \top$
$\Rightarrow_{SysS}$	

Fig. 7. Modal tautologies for simplification

3.3 Semantic Branching

The technique of semantic branching allows us to make sure that created sub-proofs are strictly disjoint. This prevents duplicate proofs a formulas which might otherwise appear in both sub-proofs. This technique may however enlarge the proof tree.

Propositional Semantic Branching In the propositional case, there is just one branching rule, the conjunction rule. A semantic version of this rule is shown

in Figure 8. The justification for replacing the standard conjunction rule by the rule for semantic conjunction branching is simple:

Lemma 1. *A sequent is provable by syntactic propositional branching iff it is provable by semantic propositional branching.*

Proof. This fact follows from the propositional tautology

$$(A \wedge B) \leftrightarrow (A \wedge (A \rightarrow B)).$$

Syntactic branching	Semantic branching
$\dfrac{\Gamma, A \qquad \Gamma, B}{\Gamma, A \wedge B}$	$\dfrac{\Gamma, A \qquad \Gamma, \neg A, B}{\Gamma, A \wedge B}$

Fig. 8. Syntactic vs. Semantic branching on conjunctions

Example 1. Consider the sequent $\Gamma = \{A \wedge B, C \wedge A\}$ and let A be provable, but let A induce a lengthy proof tree. The use of syntactic branching leads to the following redundant proof tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{\checkmark}{\dots}}{A} \quad \dfrac{\dfrac{\checkmark}{\dots}}{A, C}}{A, A \wedge C} \qquad \dfrac{\dfrac{\dfrac{\checkmark}{\dots}}{B, A} \quad \dfrac{\dots}{B, C}}{B, A \wedge C}}{A \wedge B, A \wedge C}$$

Semantic branching on the other hand may lead to a much smaller proof tree (since A has to be shown only once):

$$\dfrac{\dfrac{\dfrac{\checkmark}{\dots}}{A} \quad \dfrac{\checkmark}{A, \neg A, C}}{A, A \wedge C} \qquad \dfrac{\dfrac{\checkmark}{\neg A, B, A} \quad \dfrac{\dots}{\neg A, B, C}}{\neg A, B, A \wedge C}}{A \wedge B, A \wedge C}$$

Modal Semantic Branching The general principle that can be extracted from propositional semantic branching is a follows:

A syntactic branching rule

$$\dfrac{\Gamma_1 \qquad \dots \qquad \Gamma_i \qquad \dots \qquad \Gamma_n}{\Gamma}$$

may be (repeatedly) replaced by a 'more semantic' branching rule

$$\frac{\Gamma_1 \quad \ldots \quad \Gamma_i' \quad \ldots \quad \Gamma_n}{\Gamma}$$

where $\Gamma_i' = \bigwedge_{t \in T} \Gamma_t \to \Gamma_i$ for some $T \subseteq \{1, \ldots, n\} \setminus \{i\}$.

The justification for this generalized semantic branching is as simple as in the propositional case:

Lemma 2. *A sequent is provable by syntactic modal branching iff it is provable by semantic modal branching.*

Proof. This fact follows from the propositional tautology

$$(\textstyle\bigwedge_{t \in T} \Gamma_t \wedge \Gamma_i) \leftrightarrow (\bigwedge_{t \in T} \Gamma_t \wedge (\bigwedge_{t \in T} \Gamma_t \to \Gamma_i)).$$

Feature	Semantic Branching
$\Diamond^i_{\mathbf{G}}$	
$\ldots$	$\ldots$
$\Rightarrow_{\mathbf{CK}}$	$\dfrac{A_0 \leftrightarrow \ldots \leftrightarrow A_n \qquad \neg(A_0 \leftrightarrow \ldots \leftrightarrow A_n), \neg B_1, \ldots, \neg B_n, B_0}{\Gamma, \bigwedge_{i=1}^{n} (A_i \Rightarrow_{\mathbf{CK}} B_i) \to (A_0 \Rightarrow_{\mathbf{CK}} B_0)}$

Fig. 9. Specific semantic branching for examplary modal logics

3.4 Controlled Backtracking

The technique of *controlled backtracking* (also known as dependency directed backtracking) is in fact a special instance of caching. When this optimisation is employed, each sequent Γ depends on a set $\sharp(\Gamma)$ of sequents (namely on those sequents which introduced any formula $\phi \in \Gamma$ into the proof tree). Whenever a sequent turns out to be provable, its provability depends on possibly several sequents. These sequents are called the respective *dependency set* of $\{\top\}$, denoted by $\sharp(\{\top\})$ (these sequents are also known as the *promotors* of the successful leaf).

We have yet to formally define dependency sets, but we will rely on the following crucial fact:

Lemma 3. *Given a dependency set $\sharp(\{\top\})$ of a succesfull leaf, any sequent Γ s.t. $\Gamma \in \sharp(\{\top\})$, is provable.*

Due to Lemma 3, it will not be necessary to treat any open sequents between the current successful node in the proof graph and the first node which is contained in the according dependency set.

To achieve this, a set of *current* dependency sequents is used, and whenever a sequent is provable, the current set of dependency sequents is defined as the set of dependency sequents of $\{\top\}$. Now all those open sequents which appear during the following backtracking through the proof tree and which are created by a sequent which is *not* in the current set of dependency states, may be directly set to *provable*.

If however a node in the proof graph which is contained in the current set of dependency states is reached, the current set of dependency states is set to $\emptyset$ and the proof may continue as usual.

Definition 1. *Depending on the shape of a proof tree, $\sharp(\Gamma)$ is defined as follows:*

- *If Γ is the root of the proof tree, $\sharp(\Gamma) = \Gamma$.*
- *If Γ is obtained from Γ' by means of normalisation or simplification, then $\sharp(\Gamma) := \sharp(\Gamma')$.*
- *If Γ is the only sequent in the premise of a rule application of a non-branching rule to Γ', then $\sharp(\Gamma) := \sharp(\Gamma')$*
- *If $\Gamma_1, \ldots, \Gamma_n$ are n sequents from the premise of the rule application of a branching rule to Γ', the dependancy set of any formula $\phi \in \bigcup_{i\in\{1,\ldots,n\}} \Gamma_i$ is defined as follows: if $\forall i \in \{1, \ldots, n\}.\phi \in \Gamma_i$, then $\sharp(\{\phi\}) := \sharp(\Gamma)$, else $\sharp(\{\phi\}) := \Gamma$.*
- *Finally, $\sharp(\Gamma_1 \cup \Gamma_2) := \sharp(\Gamma_1) \cup \sharp(\Gamma_2)$*

Feature	Modal Rule	Dependencies
$\Box_{\mathbf{K}}$	$\dfrac{\overbrace{\bigwedge_{i=1}^{n} A_i \to B}^{\equiv:\Gamma_1}}{\Gamma, \underbrace{\bigwedge_{i=1}^{n} \Box_{\mathbf{K}} A_i \to \Box_{\mathbf{K}} B}_{\equiv:\Gamma'}}$	$\sharp(\Gamma_1) := \sharp(\Gamma')$
$\Rightarrow_{\mathbf{CK}}$	$\dfrac{\overbrace{A_0 \leftrightarrow \ldots \leftrightarrow A_n}^{\equiv:\Gamma_1} \quad \overbrace{\neg B_1, \ldots \neg B_n, B_0}^{\equiv:\Gamma_2}}{\Gamma, \underbrace{\bigwedge_{i=1}^{n} (A_i \Rightarrow_{\mathbf{CK}} B_i) \to (A_0 \Rightarrow_{\mathbf{CK}} B_0)}_{\equiv:\Gamma'}}$	$\sharp(\Gamma_1, \Gamma_2) := \Gamma'$

Fig. 10. Propagation of dependencies over some examplary modal features

Inituitively, a single application of for instance the modal rule of **K** does not *choose* any subformulas of its conclusionary sequent to generate new sequents (this is already a consequence of the fact that this rule does not branch). Hence

any introduced formulas will depend on those formulas on which the conclusionary sequent already depended. On the other hand, the modal rule of **CK** selects some subformulas (the antecedents) from its conclusionary sequent to be in the first sequent of the premise and others (the consequents) to be in the second sequent of the premise, thus a choice is made and the introduced formulas will directly depend on the conclusionary sequent.

The following Lemmas establish that dependency directed backtracking as defined above is a sound optimisation technique for any coalgebraic model logic:

Lemma 4. *Assume an application of any (possibly branching) rule:*

$$\frac{\Gamma_1 \quad \dots \quad \Gamma_n}{\Gamma}$$

Assume a sequent Γ' such that $\Gamma \notin \sharp(\Gamma')$, but $\Gamma' \in \Gamma_i$ for any $0 < i \leq n$. Then provability of Γ' implies the provability of all sequents of the premise of the rule.

Proof. Let Γ' be provable. Since $\Gamma \notin \sharp(\Gamma')$, we have a non-selecting rule application (w.r.t. Γ'). For non-branching rules, the implication is trivial. For a branching rule, it follows directly from the definition of $\sharp(\_)$ and the fact that one Γ_j contains Γ', that all Γ_j contain Γ'. Hence all the sequents in the premise of the rule are provable.

Lemma 5. *Let the following be a (not yet fully explored) proof tree for Γ (where Γ_j^i denotes the sequent number j on level i; in case that $j > 1$ and $i > 1$, Γ_j^i is assumed to be an open sequent):*

$$\dfrac{\dfrac{\dfrac{\dfrac{\checkmark}{\Gamma_1^n}}{\dots}}{\Gamma_1^2} \quad \dots \quad \Gamma_i^2}{\Gamma_1^1} \quad \dots \quad \Gamma_j^1}{\Gamma}$$

Further, let $\sharp(\{\top\})$ denote the dependency set of the successful leaf. If for all Γ_1^i with $0 < i \leq n$, $\sharp(\{\top\}) \not\subseteq \Gamma_1^i$, and if $\sharp(\{\top\}) \subseteq \Gamma$ (i.e. Γ is the first sequent contained in the dependency set of the successful leaf), then Γ_1^1 is provable (since any Γ_j^i with $i > 1$ is provable).

Proof. Lemma 4 ensures, that the provability of the successful leaf propagates to all the open sequents Γ_j^i for $i, j > 1$ (since $\sharp(\{\top\}) \not\subseteq \Gamma_j^{i-1}$). Thus we are not able to find any sequent of level > 1 that is not provable.

To summarize, once we found a successful leaf, it is not necessary to treat the whole sub-proof which is induced by Γ_1^1, since Γ_1^1 is always provable in the above situation.

3.5 Maximal Application of Modal Rules

A single modal sequent rule representation usually describes in fact a set of modal rules, each of the rules treating a different combination of positive and/or negative literals of the respective feature. Given a sequent Γ to match, any such modal rule could be applied to Γ, as long as the conclusion of the rule is equal to Γ. It may be observed however that it often suffices to consider *maximal* applications of the modal rules. An application (Γ_1, Γ_1') of a modal rule to $\Gamma_1 \subseteq \Gamma$ is maximal if there is no formula ϕ in $\Gamma \setminus \Gamma_1$ s.t. the conclusion of any the modal rule would still be of shape $\Gamma_1 \cup \{\phi\}$.

Given a set of modal rules $\mathcal{R}_M$, the restriction to maximal applications of rules from $\mathcal{R}_M$ is admissable whenever the provability of the premise of any non-maximal application of a rule from $\mathcal{R}_M$ implies that there is a maximal application of a rule $\mathcal{R}_M$ with a provable premise.

Such a restriction is admissable (and in fact highly recommended) for many modal logics (see Table 11)

A negative example is the feature of **CK** (or similarly, the feature of **CK-CEM**):

Example 2. Let us consider the examplary sequent $\Gamma = \{\neg(A_1 \Rightarrow B_0), \neg(A_2 \Rightarrow B_0), (A_0 \Rightarrow B_0)\}$ and assume that $A_0 \leftrightarrow A_1$ but $A_1 \not\leftrightarrow A_2$. The only maximal application of the modal rule **CK** would lead to three new sequents:

$$\dfrac{\dfrac{\checkmark}{A_0 \leftrightarrow A_1} \qquad \dfrac{\lightning}{A_1 \leftrightarrow A_2} \qquad \dfrac{\checkmark}{\neg B_0, B_0}}{\neg(A_1 \Rightarrow B_0), \neg(A_2 \Rightarrow B_0), (A_0 \Rightarrow B_0)}$$

The second of these sequents is not provable by assumption, thus ϕ can not be shown to be provable.

However, using a non-maximal application of the modal rule **CK** to the subsequent $\Gamma' = \{\neg(A_1 \Rightarrow B_0), (A_0 \Rightarrow B_0)\}$ of Γ, the provability of ϕ may be shown:

$$\dfrac{\dfrac{\checkmark}{A_0 \leftrightarrow A_1} \qquad \dfrac{\checkmark}{\neg B_0, B_0}}{\neg(A_1 \Rightarrow B_0), \neg(A_2 \Rightarrow B_0), (A_0 \Rightarrow B_0)}$$

Even though the restriction to maximal applications of the modal rule of **CK** is not admissable, it is possible to obtain a nearly equivalent optimisation of the proofs for **CK** by the use of intelligent modal heuristics (see 3.8).

3.6 Proof-tree Trimming

Due to the two-kinded nature of a proof tree and the alternating existential and universal provability requirements between the two kinds of nodes, a proof

Modal Features	Admissability
$\Box_{\mathbf{K}}, \Box_{\mathbf{KD}}, \Box_{\mathbf{M}}, \Box^{a}_{\mathbf{HM}}, \Diamond^{i}_{\mathbf{G}}, \Diamond^{p}_{\mathbf{P}}, \Rightarrow_{\mathbf{CK_{CM}}}, \Rightarrow_{SysS}$	Yes
$\Rightarrow_{\mathbf{CK}}, \Rightarrow_{\mathbf{CK_{CEM}}}$	No

Fig. 11. Admissability of restriction to maximal application of modal rules

tree may be trimmed, once enough information about the provability of specific sequents has been obtained.

If there are several possible premises for one sequent and the provability of one of the premises has already been shown, the other premises for the same sequent need not be explored any more. Similary, if there are several sequents in a premise, as soon as one of the sequents turns out to be not provable the whole premise is not provable any more and the other sequents of the premise need not be explored.

In more detail, the following operations are valid trimmings of a proof tree:

1. Removal of non-provable premises,
2. Removal of all premises that are parallel to a provable premise ($r_1(\Gamma) = \{\Sigma_1, \ldots, \Sigma_i, \ldots, \Sigma_n\}$ and Σ_i provable $\rightsquigarrow r_1(\Gamma) = \{\Sigma_i\}$),
3. Removal of provable sequents,
4. Removal of all sequents that are parallel to a non-provable sequent ($r_2(\Sigma) = \{\Gamma_1, \ldots, \Gamma_i, \ldots, \Gamma_n\}$ and Γ_i non-provable $\rightsquigarrow r_2(\Sigma) = \{\Gamma_i\}$).

The resulting tree is no complete proof tree any more (since several branches may have been cut of). However, the following fact establishes that trimming is a sound optimisation technique:

Lemma 6. *A sequent Γ has a successful proof tree iff it has a successful trimmed proof tree.*

Proof. The branches which are cut off do not interfere with the provability of Γ. See 4.3 for more details.

3.7 Caching

Using the optimisation technique of caching, it is possible to store the satisfiability status of subformulas that appear during the course of a proof. This prevents us from treating the same subformula more than once, thus saving redundant proof-work.

The cost of this optimisation technique is an increase in the use of memory during the proof. Especially in the case of logics whose decision procedure is in PSPACE, the use of caching may in theory negate the property of optimal complexity of the used algorithm (bringing those logics in theory to EXPTIME).

Caching is a bit troublesome in combination with dependency directed backtracking since a formula may depend on several sequents if it appears at different positions in the proof. However, it is sufficient to simply use the union of the respective dependency sets whenever a formula appears at least twice.

3.8 Heuristics

A heuristic guides the search in the proof tree and chooses

1. a premise from the set of all open premises,
2. a meta-disjunction within this premise and
3. one sequent within the selected meta-disjunction.

To this end, the heuristic assigns a priority to each open premise, each open meta disjunction and each open sequent.

The sequent with the highest overall priority will be the next to be explored.

A reasonable heuristic helps (if possible) in finding a shorter way to show provability of a formula.

Propositional expansion first It is a reasonable tactic of expansion to first completely treat all propositional formulas in a sequent before considering the modal literals. This may be achieved by assigning high priority to those sequents, meta-disjunctions and premises which are introduced by the conjunction rule. Such sequents, meta-disjunctions and premises which are generated by an application of a modal rule are given a low priority.

Classic Heuristics On top of this preference of propositional expansion, one might add any other heuristic, such as for instance one of the following two classic heuristics (or even a combination of both):

- *most appearing first*: Assign the highest priority to open sequents that appear in the highest number of formulas. The priority of meta-disjunctions is defined to be the priority of the contained sequent. The priority of a premise is the maximum of the priorities of the contained meta-disjunctions.
- *oldest first*: ...

Modal Heuristics Here is a specific modal heuristic for **CK** (a similar heuristic if suitable for **CKCEM**). Note, that this heuristic is only applicable if caching is enabled:

Within any premise of a modal rule, the short sequents of the form $A \leftrightarrow B$ are given the highest priority (in combination with caching, this leads to a complete partition of modal antecedents into equivalence classes).

Once this is done, we give high priority (e.g. 1) to *maximal* (w.r.t the cached equivalence classes) premises and priority of 0 to non-maximal premises (i.e. we do not treat non-maximal premises at all). As long as not all significant equivalences have yet been shown or refuted, the priority of a premise stays at a normal value (e.g. 0.5).

In combination with caching (of logical equivalences), this heuristic implements the optimizations of **CK** and **CKCEM** proposed in Hausmann,Schröder 2009.

4 Global Caching Algorithm

This is an adoptation of the Algorithm from Pattinson et.al.

Key differences are

1. We treat provability of formulas here instead of satisfiability. However, a formula is satisfiable whenever its negation is not provable.
2. The introduction of meta-disjunctions.
3. X is a structured set now. This allows heuristics to first choose a premise to treat, then choose a meta-disjunction from this premise and finally choose a sequent from the chosen meta-disjunction.
4. The propagation is optimized now. Instead of computing the fixpoints of all the undecided sequents every time, classes of coherent sequents are stored. The propagation checks only 'whether the new sequents decide any of the classes'.

4.1 The proof graph and transitions

A sequent is a set of formulas. A meta-disjunction is a set of sequents. A premises is a set of meta-disjunctions.

A graph is a tuple (A, E, U, X, L_1, L_2) where $A, E, U \subseteq Seq$ are sets of sequents, $X \subseteq Prems$ is a set of premises.

Given a sequent Γ, we define the set of all possible rule applications to Γ, $AR(\Gamma) = \{(\Gamma, \Sigma) \in \mathcal{R}\}$, the set of all obtainable premises $AP(\Gamma) = \{\Sigma \mid (\Gamma, \Sigma) \in AR(\Gamma)\}$ and the set of all selections $AS(\Gamma) = \{(\Sigma, \Gamma') \mid \Sigma \in AP(\Gamma) \wedge \Gamma' \in \Sigma\}$.

We fix two effectively computable transitions between graphs:

- *Expand:* Given a sequent Γ that is contained in any meta-disjunction from any premise in X, this transition leads from a graph g to the graph g' that is obtained from g by expanding Γ. To this end, X is extended by all the premises from $AR(\Gamma)$ and Γ is removed from X ($X' = (X \cup AP(\Gamma)) \setminus \Gamma$). Furthermore, L_1 is extended by all possible rule applications for Γ ($L_1' = L_1 \cup AR(\Gamma)$). Finally, L_2 is extended by all possible selections for rule applications for Γ ($L_2' = L_2 \cup AS(\Gamma)$).
  In conclusion, this procedure applies all applicable rules to Γ, marks all resulting premises as not yet expanded, marks Γ as expanded and stores the implied transitions in L_1 and L_2.
  For a fixed sequent Γ and a graph g, we denote the expansion of Γ in g by $exp(g, \Gamma)$. If $g' = exp(g, \Gamma)$, we also write $g \xrightarrow{\Gamma}_E g'$.
- *Propagate:* This procedure evaluates the given graph as far as yet possible: The sets X, L_1 and L_2 remain unchanged. The set of provable sequents however is extended by the least fixpoint of a function M^L ($A' = A \cup \mu(M^L)$). The set of not provable sequents is extended by the greatest fixpoint of a function W^L ($E' = E \cup \nu(W^L)$). Finally, those sequents that are known to be either provable or not provable are removed from the set of undecided

sequents ($U' = U \setminus (A' \cup E')$).

The functions which are used for the fixpoint computations are defined as follows:

$$M^L(X) = \{\Gamma \in U \mid \forall(\Gamma, \Sigma) \in L_1.\exists(\Sigma, \epsilon) \in L_2.\forall\Gamma' \in \epsilon.\Gamma' \in X \cup A\}$$
$$W^L(X) = \{\Gamma \in U \mid \exists(\Gamma, \Sigma) \in L_1.\forall(\Sigma, \epsilon) \in L_2.\exists\Gamma' \in \epsilon.\Gamma' \in X \cup E\}$$

Given a graph g, we denote the graph obtained by propagating in g by $prp(g)$. If $g' = prp(g)$, we also write $g \rightarrow_P g'$.

4.2 The Algorithm

In order to show the provability of a formula ϕ, we start off with the sequent $\Gamma_0 = \{\phi\}$:

Example 3. (Show provability of Γ_0)

1. Set $g = (A = \{\emptyset\}, E = \emptyset, U = \{\Gamma_0\}, X = AP(\Gamma_0), L_1 = AR(\Gamma_0), L_2 = AS(\Gamma_0))$.
2. If $X = \emptyset$, return the current graph as result.
3. Otherwise select any premise Σ from X, select any meta-disjunction ϵ from Σ and select any sequent Γ from ϵ. Compute $g' = exp(g, \Gamma)$ s.t. $g \xrightarrow{\Gamma}_E g'$. (Optionally: Compute $g'' = prp(g')$ s.t. $g' \rightarrow_P g''$; If $\Gamma_0 \in A'' \cup E''$, break and return the current graph as result.)
4. Set the current graph to g' (or g'' if a propagation step took place). Continue with 2.

After the loop has finished, propagate one final time and then let A (E) be the set of provable (not provable) sequents of the resulting graph. If $\Gamma_0 \in A$, return `True`, if $\Gamma_0 \in E$, return `False`, otherwise return `Undecided`.

This algorithm has been shown to be

a) sound and complete w.r.t. provability of formulas in the logic that is represented by the utilized set of rules $\mathcal{R}$,
b) of complexity EXPTIME, if the underlaying ruleset allows for it.

4.3 Improving the algorithm

Due to the inherent inefficiency of computing the propagation of A (E) through the proof graph, we devise the following more efficient method:

Definition 2. *A* position $p \in Pos = \{c_1 \dots c_n \mid c_1, \dots, c_n \in \mathbb{N}\} \cup \{[]\}$ *is a finite list of natural numbers.*

A proof formula *is defined as follows (for* $j, p \in Pos$*):*

$$\mathcal{PF} \ni \psi_1, \dots, \psi_n := a_j \mid \wedge\{\psi_1, \dots, \psi_n\} \mid \vee\{\psi_1, \dots, \psi_n\} \mid l_p$$

The subformula $\phi|_p$ of a proof formula ϕ at position $p = c_1 \dots c_n$ is defined recursively. If $n = 0$, then $\phi|_p = \phi$. If $n = 1$ (i.e. $p = c_1$), then $p_2 = []$, otherwise (i.e. $n > 1$, $p = c_1 c_2 \dots c_n$), $p_2 = c_2 \dots c_n$.

- *If $n > 0$, $a_j|_p$ and $l_i|_p$ are undefined.*
- *Let $\phi = \wedge\{\psi_1, \dots, \psi_n\}$ or $\phi = \vee\{\psi_1, \dots, \psi_n\}$. If $0 < c_1 \leq n$, $\phi|_p = \psi_{c_1}|_{p_2}$; if $c_1 > n$, $\phi|_p$ is undefined.*

The partial indexing function $(\!|\_|\!) : Seq + Prem \to Pos$ *maps sequents $\Gamma \in Seq$ or premises $\Sigma \in Prem$ from the defined subset of its domain to a position.*

We also introduce sets of provable positions $A_p \subseteq Pos$ and non-provable positions $E_p \subseteq Pos$ as well as the set tcs of positions that have been expanded since the last propagation step and that are relevant to the propagation.

When constructing the proof graph g for a formula ϕ, we build up a proof formula $pf(g)$ in parallel, beginning with $pf(g_0) = a_{(\!|\{\phi\}|\!)}$, $(\!|\_|\!)$ undefined except for $(\!|\{\phi\}|\!) = []$ and $A_p = E_p = tcs = \emptyset$.

Expanding the proof formula: Say we expand sequent Γ, s.t. $g \xrightarrow{\Gamma}_E g'$ for $g' = exp(g, \Gamma) = (A', E', U', X', L_1', L_2')$. This means that there are n premises to Γ (i.e. $|AP(\Gamma)| = n$) and for each $\Sigma_i \in AP(\Gamma)$, $(\Gamma, \Sigma_i) \in L_1'$. Also, for each $\Sigma_i \in AP(\Gamma)$, there is an m_i s.t. $|\{\Gamma_j \mid (\Sigma_i, \Gamma_j) \in AS(\Gamma)\}| = m_i$ and each such (Σ, Γ_j) is contained in L_2'. In other words, Γ has n premises, each premise consists of m_i sequents and the according transitions are stored in L_1' and L_2'.

The according changes to the proof formula and the indexing function are as follows:
First we expand the sequent Γ:

- Replace $a_{(\!|\Gamma|\!)}$ with $\vee\{b_1, \dots, b_n\}$.
- For $0 < i \leq n$, if $(\!|\Sigma_i|\!)$ is defined (i.e. the premise was encountered before), replace b_i by $l_{(\!|\Sigma_i|\!)}$. If $(\!|\Sigma_i|\!) \in A_p \cup E_p$, add $(\!|\Gamma|\!)$ to tcs.
- For each remaining b_i (i.e. for each new premise), set $(\!|\Sigma_i|\!) = (\!|\Gamma|\!)i$.
- *Optional:* If $n = 1$, then $\vee\{b_1\} = b_1$ and $(\!|\Sigma_1|\!) = (\!|\Gamma|\!)$.
- If $n = 0$, then add $(\!|\Gamma|\!)$ to tcs.

Then we expand all the premises Σ_i:

- Replace any b_i with $\wedge\{a_1, \dots, a_{m_i}\}$.
- For $0 < j \leq m_i$, if $(\!|\Gamma_j|\!)$ is defined (i.e. the sequent was encountered before), replace a_j by $l_{(\!|\Gamma_j|\!)}$. If $(\!|\Gamma_j|\!) \in A_p \cup E_p$, add $(\!|\Sigma_i|\!)$ to tcs.
- For each remaining a_j (i.e. for each new sequent), set $(\!|\Gamma_j|\!) = (\!|\Sigma_i|\!)j$.
- *Optional:* If $m_i = 1$, then $\wedge\{a_1\} = a_1$ and $(\!|\Gamma_1|\!) = (\!|\Sigma_i|\!)$.
- If $m_i = 0$, then add $(\!|\Sigma_i|\!)$ to tcs.
- Replace any a_j with $a_{(\!|\Gamma_j|\!)}$.

Propagation in the proof formula: Propagation of A (E) through the proof formula is realized by iterative approximation of the fixpoints. Any of the rules from Figure 12 is applied repeatedly for each $pj \in tcs$ until no rule may be applied for this specific pj any more, pj is removed from tcs afterwards:

Condition(s)	Changes to A_p (E_p)	Changes to tcs
$\phi\|_{pj} = \wedge\emptyset$	add pj	
$\phi\|_{pj} = \vee\emptyset$	(add pj)	
$\exists q \in Pos.\phi\|_q = l_{pj}$, $pj \in A_p$	add q	add q
$\exists q \in Pos.\phi\|_q = l_{pj}$, $pj \in E_p$	(add q)	add q
$\phi\|_p = \wedge\{\psi_1, \ldots, \psi_n\}$, $\forall i \in \{1 \ldots n\}.pi \in A_p$	add p	add p
$\phi\|_p = \vee\{\psi_1, \ldots, \psi_n\}$, $\exists i \in \{1 \ldots n\}.pi \in A_p$	add p	add p
$\phi\|_p = \wedge\{\psi_1, \ldots, \psi_n\}$, $\exists i \in \{1 \ldots n\}.pi \in E_p$	(add p)	add p
$\phi\|_p = \vee\{\psi_1, \ldots, \psi_n\}$, $\forall i \in \{1 \ldots n\}.pi \in E_p$	(add p)	add p

Fig. 12. The simplification rules for propagation in proof formulas

Propagation adds the positions of all newly added empty conjunctions (and disjunctions) to A_p (E_p) and propagates them through the formula. The set tcs is empty after each completed propagation step.

The set of all atoms a_j that appear in $pf(g)$ is the relevant subset of the set X of unexpanded sequents of the proof graph.

Lemma 7. $\{\Gamma \in Seq \mid (\!|\Gamma|\!) \in A_p\} = A$, $\{\Gamma \in Seq \mid (\!|\Gamma|\!) \in E_p\} = E$.

Proof. The constructed proof formula is a direct representation of the proof graph. Repeated application of the rules from Figure 12 iteratively approximates the fixpoint A (E) and stores the according positions in A_p (E_p).

For more details, see the appendix.

It follows that ϕ is provable iff $[] \in A_p$ and that if $[] \in E_p$, ϕ is not provable.

Simplification: The following are valid methods for the simplification of the proof formula:

Contract links correctly.

If $\psi_i = \psi_j$, then $\wedge\{\psi_i, \psi_j, \ldots\}$ may be replaced by $\wedge\{\psi_i, \ldots\}$, remove j from I_k. If $\psi_i = \psi_j$, then $\vee\{\psi_i, \psi_j, \ldots\}$ may be replaced by $\vee\{\psi_i, \ldots\}$, remove j from I_k. If $\phi_i = \psi_j$, $\psi_h = \wedge\{\phi_i, \phi_1, \ldots, \phi_n\}$, then $\vee\{\psi_h, \psi_j, \ldots\}$ may be replaced by $\vee\{\psi_h, \ldots\}$, remove j from I_k. If $\phi_i = \psi_j$, $\psi_h = \vee\{\phi_i, \phi_1, \ldots, \phi_n\}$, then $\wedge\{\psi_h, \psi_j, \ldots\}$ may be replaced by $\wedge\{\psi_j, \ldots\}$, $I_{k1} = \emptyset$, remove h from I_k.

A homogenous disjunctive circle $\vee\{\vee\{\ldots\{\vee\{n_p, \ldots\}, \ldots\}\ldots\}, \ldots\}$ of length j starting at position p may be replaced by $\vee\{\psi_i \mid i \in I\} \cup \{n_p\}$ where $I = \bigcup\{I_k \mid k = pl\}$ such that l is a position within the circle. Any reference to the removed positions must be set to p.

A homogenous conjunctive circle $\wedge\{\wedge\{\ldots\{\wedge\{n_p, \ldots\}, \ldots\}\ldots\}, \ldots\}$ of length j starting at position p may be replaced by $\wedge\{\psi_i \mid i \in I\} \cup \{n_p\}$ where $I = \bigcup\{I_k \mid k = pl\}$ such that l is a position within the circle. Any reference to the removed positions must be set to p.

Notice that additionally to being more efficient than the previous variant, the new methodology allows for a nice heuristic:

– Only those sequents Γ which have a representation in the current proof formula need to be explored (i.e. for Γ to be expanded, $(\!|\Gamma|\!)$ must be a valid position in the current proof formula).

What about the unexpanded sequents which have no representation in the proof formula? They either appear as sequent in a premise that already contains one non provable sequent (s.t. the premise itself is not provable), or they are sequents from a premise that is no longer needed to be shown to be provable (since there exists already another provable premise for the same sequent). In both cases, the provability (or non-provability) of the according sequents does not influence the overall outcome of the proof.

4.4 Removing Caching

What if we propagate after every step and always 'empty' the graph? Does it remove the caching while staying sound and complete and does it lead to PSPACE?

5 Implementation

...

6 Conclusion

...

7 Appendix (& old stuff)

Definition 3. *A* proof formula *is defined as follows for* $i, j \in \mathbb{N}$*:*

$$\mathcal{GF} \ni \psi, \phi := a_j \mid \top \mid \bot \mid \psi \wedge \phi \mid \psi \vee \phi \mid n_i$$

We also assign a number $i \in \mathbb{N}$ *(its index) to any proof formula* ϕ *that we use and henceforth denote this formula by* ϕ^i*. An indexing function* $(\!|\_|\!) : (Seq + Prem) \to \mathbb{N}$ *returns the index of the representing proof formula for either a sequent* Γ *or a premise* Σ*. This function is assumed to be initially undefined for any sequent or premise (except for* $\{\phi\}$*).*

A graph formula ϑ *for an index set* I *is just a set* $\vartheta = \{\psi^i \mid i \in I\}$*. A* wellformed graph formula ϑ *for an index set* I *is a graph formula s.t. for any node operator* n_j *that appears in any of the* ψ^i*,* $j \in I$*. W.l.o.g. we assume that there is one* $\psi^{(\!|\{\phi\}|\!)}$ *representing our initial sequent* $\{\phi\}$ *and that all other* ψ^j *are reachable from* $\psi^{(\!|\{\phi\}|\!)}$*.*

Our intuition of a graph formula will be that for truth, falsedom, conjunction and disjunction it behaves as usual, and the semantics of a node operator n_i is just the semantics of the formula that the node points to (i.e. ψ^i). An atom a_j will represent the provability status of a sequent, i.e. if a sequent is known to be provable (not provable), the according atom will in the end evaluate to $\top$ ($\bot$).

To achieve this, we assign two graph formulas $gf_1(g) = \{\psi_1^1, \ldots, \psi_1^n\}$ and $gf_2(g) = \{\psi_2^1, \ldots, \psi_2^m\}$ to any proof graph $g = (A, E, U, X, L_1, L_2)$ as follows:

Given a formula ϕ, we initialize $gf_1(g_0) = gf_2(g_0) = \{a_{(\!|\{\phi\}|\!)}\}$ and continue to expand the graph formula in step with our expansion of the proof graph. We also initialize the sets $A_{seq}(g_0) = \emptyset$, $E_{seq}(g_0) = \emptyset$, $A_{prem}(g_0) = \emptyset$ and $E_{prem}(g_0) = \emptyset$.

Say we expand sequent Γ, s.t. $g \xrightarrow{\Gamma}_E g'$ for $g' = exp(g, \Gamma) = (A', E', U', X', L_1', L_2')$. This means that there are n premises to Γ (i.e. $|AP(\Gamma)| = n$) and for each $\Sigma \in AP(\Gamma)$, $(\Gamma, \Sigma) \in L_1'$. Also, for each $\Sigma \in AP(\Gamma)$, there is an m s.t. $|\{\Gamma' \mid (\Sigma, \Gamma') \in AS(\Gamma)\}| = m$ and each such (Σ, Γ') is contained in L_2'. In other words, Γ has n premises, each premise consists of a number of sequents and the according transistions are stored in L_1' and L_2'.

Then the expansion of the first graph formula is realized as follows (where we obtain a new graph formula $gf_1'(g)$ from $gf_1(g)$ by substituting parts of the graph formula):

1. If $n = 0$, then there is no rule that may be applied to Γ, i.e. Γ is not provable. The expanded graph formula is obtained substituting $a_{(\!|\Gamma|\!)}$ with $\bot$.
2. If $n = 1$, then there is just one rule that may be applied to Γ. This allows us to directly continue without adding a new proof formula, that is, we replace $a_{(\!|\Gamma|\!)}$ with b_Σ for the one $\Sigma \in AP(\Gamma)$ s.t. b_Σ represents $\Sigma \in AP(\Gamma)$ and is defined as follows:
   - if $(\!|\Sigma|\!)$ is yet undefined (i.e. the premise does not yet appear in the proof graph), then $b_\Sigma = \psi^{(\!|\Gamma|\!)}$ and $(\!|\Sigma|\!) := (\!|\Gamma|\!)$, else
   - $b_{\Sigma_j} = n_{(\!|\Sigma_j|\!)}$.
3. If $n > 1$, we globally substitute $a_{(\!|\Gamma|\!)}$ by the *disjunction* of all b_{Σ_j} where each b_{Σ_j} stands for the premise of one of the n possible rule applications to Γ, s.t. b_{Σ_j} represents $\Sigma_j \in AP(\Gamma)$ and is defined as follows:
   - if $(\!|\Sigma_j|\!)$ is yet undefined (i.e. the premise does not yet appear in the proof graph), then $b_{\Sigma_j} = n_k$ for a new $k \notin I$, and set $(\!|\Sigma_j|\!) = k$, else
   - $b_{\Sigma_j} = n_{(\!|\Sigma_j|\!)}$.

We continue with the following case distinction for each new ψ^k:

1. If the according $m = 0$, then the premise contains no sequent and is hence provable. The according ψ^k is directly defined as $\top$.
2. If $m = 1$, then Σ_j contains just one sequent. This allows us to directly continue without adding a new proof formula. This means that ψ^k is defined as follows (for the one Γ' s.t. $(\Sigma_j, \Gamma') \in L_2'$).
   - If $(\!|\Gamma'|\!)$ is yet undefined (i.e. the sequent does not yet appear in the proof graph), then $\psi^k = a_{(\!|\Sigma_j|\!)}$ and $(\!|\Gamma'|\!) := (\!|\Sigma_j|\!)$, else
   - $\psi^k = n_{(\!|\Gamma'|\!)}$.

3. If $m > 1$, the new formula ψ^k consists of the *conjunction* of all the m atoms a_{Γ_l} for sequents from Σ_j (where $\Gamma_l \in \{\Gamma' \mid (\Sigma_j, \Gamma') \in L_2'\}$). Each a_{Γ_l} is defined as follows:
   - if $(\!|\Gamma_l|\!)$ is yet undefined (i.e. the sequent does not yet appear in the proof graph), then $a_{\Gamma_l} = n_k$ for a new $k \notin I$ and set $(\!|\Gamma_l|\!) = k$, else
   - $a_{\Gamma_l} = n_{(\!|\Gamma_l|\!)}$.

Finally, for every n_k that was added in the last step, add a new proof formula ψ_k to the graph formula, s.t. $\psi_k = a_{(\!|\Gamma_l|\!)}$.

This builds a graph formula $gf_1'(g) = gf_1(g')$ that completely characterizes the current state of the proof while usually being smaller than the proof graph. However, we maintain the property that the graph formula contains separate proof formulas for any different combination of sequents and premises that has yet been encountered (even though the provability of the formula in question may not depend on many of these combinations).

Definition 4. *The* contraction *of a graph formula $\phi = \{\psi_1, \ldots, \psi_n\}$, $I = \{1, \ldots, n\}$ is the graph formula that is obtained from ϕ as follows:*

> *For every $i \in I$, s.t. $\psi^i = n_j$ for some j, replace any occurance of n_i by n_j, remove ψ^i from ϕ.*

Example 4. The graph formula $gf_1(g)$ is the contraction (with contracted circles) of a graph formula that is isomorphic to g.

If we try to further reduce the size of the graph formula while preserving only the information needed to directly show (or refute) provability, we can do better. We construct an even smaller graph formula by expanding as follows:

Replace $a_{(\!|\Gamma|\!)}$ with $b_{(\!|\Sigma_1|\!)} \vee \ldots \vee b_{(\!|\Sigma_n|\!)}$ where $AP(\Gamma) = \{\Sigma_1, \ldots, \Sigma_n\}$ (if $n = 0$, replace $a_{(\!|\Gamma|\!)}$ with $\bot$). For any premise Σ_i to Γ we now distinguish three cases:

1. $\Sigma_i \in A(g)_{prem}$ ($\Sigma_i \in E(g)_{prem}$), i.e. the premise has been shown to be provable (not provable). Then replace $b_{(\!|\Sigma_i|\!)}$ with $\top$ ($\bot$).
2. $\psi^{(\!|\Sigma_i|\!)}$ is contained in $gf_1(g')$ but not in $gf_1(g)$. Then leave $b_{(\!|\Sigma_i|\!)}$ untouched.
3. $\psi^{(\!|\Sigma_i|\!)}$ is contained in $gf_1(g')$ and in $gf_1(g)$ and also $\Sigma_i \notin E(g)_{prem} \cup A(g)_{prem}$, i.e. the premise was encountered before and is not decided yet. Then take the flattening of the whole sub graph formula from $gf_1(g')$ at $(\!|\Sigma_i|\!)$ and add it to the current graph formula. Replace $b_{(\!|\Sigma_i|\!)}$ by $n_{(\!|\Sigma_i|\!)}$. Simplify this (and make any other appearance of the formula point to $(\!|\Sigma_i|\!)$).

Next, replace every remaining $b_{(\!|\Sigma_i|\!)}$ with $a_{(\!|\Gamma_1|\!)} \wedge \ldots \wedge a_{(\!|\Gamma_m|\!)}$ where Σ_i is assumed to contain just the m sequents $\Gamma_1, \ldots, \Gamma_m$ (if $m = 0$, replace the according $b_{(\!|\Sigma_i|\!)}$ with $\top$). We distinguish the cases:

1. $\Gamma_i \in A(g)_{seq}$ ($\Gamma_i \in E(g)_{seq}$), i.e. the sequent has been shown to be provable (not provable). Then replace $a_{(\!|\Gamma_i|\!)}$ with $\top$ ($\bot$).
2. $\psi^{(\!|\Gamma_i|\!)}$ is contained in $gf_1(g')$ but not in $gf_1(g)$. Then leave $a_{(\!|\Gamma_i|\!)}$ untouched.

3. $\psi^{(\!|\Gamma_i|\!)}$ is contained in $gf_1(g')$ and in $gf_1(g)$ and also $\Gamma_i \notin E(g)_{seq} \cup A(g)_{seq}$, i.e. the sequent was encountered before and is not decided yet. Then take the flattening of the whole sub graph formula from $gf_1(g')$ at $(\!|\Gamma_i|\!)$ and add it to the current graph formula. Replace $a_{(\!|\Gamma_i|\!)}$ by $n_{(\!|\Gamma_i|\!)}$. Simplify this (and make any other appearance of the formula point to $(\!|\Gamma_i|\!)$).

The constructed graph $gf_2(g)$ fully represents the current state of the proof (as stored in graph g) while being minimal.

Definition 5. *The* flattening *of a graph formula $\phi = \{\psi_1, \ldots, \psi_n\}$, $I = \{1, \ldots, n\}$ is the graph formula that is obtained from the concraction of ϕ as follows:*

> *For every $i \in I$, s.t. n_i occurs only one time in ϕ, replace n_i by ψ^i, remove ψ^i from ϕ.*

Example 5. The graph formula $gf_2(g)$ is the flattening (with flattened circles) of a graph formula that is isomorphic to g.

Contraction of circles: Whenever a link n_i to a proof formula ψ^i is added to the graph formula $gf_1(g)$ s.t. ψ^i appears in one of the paths from the formula $\psi^{(\!|\{\phi\}|\!)}$ (that represents $\{\phi\}$) to the proof formula that is currently being expanded, it may be possible to contract the circle in the graph formulas.

In more detail: Let $\psi^1, \ldots, \psi^m$ be a circle of m formulas from $gf_1(g)$, i.e. ψ^i contains at least one occurance of n_{i+1} for $0 < i < (m-1)$ and ψ^m contains n_1.

If the circle is homogenous, i.e. either for all $0 < i < m$, $\psi^i = n_j \vee \ldots \vee n_k$ (disjunctive homogenous circle) or for all $0 < i < m$, $\psi^i = n_j \wedge \ldots \wedge n_k$ (conjunctive homogenous circle), we may collapse the circle in $gf_1(g)$ as follows:

The indexing set $\mathcal{I}$ of a circle is defined as follows: $\mathcal{I} = \bigcup_{0<i<m} ind(\psi^i) \setminus \{1, \ldots, m\}$, where $ind(n_j \vee \psi) := \{j\} \cup ind(\psi)$, $ind(n_j \wedge \psi) := \{j\} \cup ind(\psi)$.

A homogenous circle may be replaced by a formula ψ^c.

For disjunctive homogenous circles, $\psi^c := n_c \vee \bigvee_{i \in \mathcal{I}} n_i$, For conjunctive homogenous circles, $\psi^c := n_c \wedge \bigwedge_{i \in \mathcal{I}} n_i$.

Replace every occurance of n_i (for $0 < i < m$) in the current graph formula by n_c.

Flattening of circles: For $gf_2(g)$, define ψ^c as the flattening of the whole sub graph formula from $gf_1(g)$ at the position i. Replace every occurance of ψ^c or n_i by n_c.

Definition 6. *Any sequent Γ (premise Σ) s.t. $i \in \wr\Gamma\int$ ($i \in \wr\Sigma\int$) is said to have a* representation *in ψ^i.*

Example 6. all.ex in graph iff contraction eval to true

Now that we defined the expansion of the two graph formulas, we describe propagation in the second graph formula, where we expand the sets $A(g)_{seq}$, $E(g)_{seq}$, $A(g)_{prem}$ and $E(g)_{prem}$:

The propagation is in fact a process of propositional simplification, executed on the current $gf_2(g)$. The following propositional tautologies (and their symmetric counterparts) are applied:

- $(\top \vee \phi) \rightarrow \top$ (only one premise needs to be provable). Also set $A(g')_{seq} = A(g)_{seq} \cup tops_{seq}$, where $tops_{seq}$ are all the sequents that have a representation in $\top \vee \phi$. Also set $A(g')_{prem} = A(g)_{prem} \cup tops_{prem}$, where $tops_{prem}$ are all the sequents that have a representation in $\top \vee \phi$.
- $(\bot \vee \phi) \rightarrow \phi$ (not provable premises are removed from the formula). Also set $E(g')_{seq} = E(g)_{seq} \cup bot$ where bot is the sequent that is represented by $\bot$. Also set $E(g')_{prem} = E(g)_{prem} \cup bot$ where bot is the premise that is represented by $\bot$.
- $(\top \wedge \phi) \rightarrow \phi$ (provable sequents are removed from the formula). Also set $A(g')_{seq} = A(g)_{seq} \cup top$ where top is the sequent that is represented by $\top$. Also set $A(g')_{prem} = A(g)_{prem} \cup top$ where top is the premise that is represented by $\top$.
- $(\bot \wedge \phi) \rightarrow \bot$ (all sequents need to be provable). Also set $E(g')_{seq} = E(g)_{seq} \cup bots$, where $bots$ are all the sequents that have a representation in $\bot \wedge \phi$. Also set $E(g')_{prem} = E(g)_{prem} \cup bots$, where $bots$ are all the premises that have a representation in $\bot \wedge \phi$.
- $(\psi^i = \top) \rightarrow (n_i \rightarrow \top)$. Trivial proof formulas may be collapsed. Also set $A(g')_{seq} = A(g)_{seq} \cup top$ where top is the sequent that is represented by $\top$. Also set $A(g')_{prem} = A(g)_{prem} \cup top$ where top is the premise that is represented by $\top$.
- $(\psi^i = \bot) \rightarrow (n_i \rightarrow \bot)$. Trivial proof formulas may be collapsed. Also set $E(g')_{seq} = E(g)_{seq} \cup bot$ where bot is the sequent that is represented by $\bot$. Also set $E(g')_{prem} = E(g)_{prem} \cup bot$ where bot is the premise that is represented by $\bot$.

As soon as $\{\phi\} \in A(g)_{seq}$ or $\{\phi\} \in E(g)_{seq}$, we are finished.

The following lemma establishes the correctness of this optimization:

Lemma 8. *Given a graph $g = (A, E, U, X, L_1, L_2)$,*

$A = A(g)_{seq}$,
$E = E(g)_{seq}$.

Proof. We treat the case of A, the case of E is dual.

First we show that A is contained in $A(g)_{seq}$.

We proceed by induction over all expansion and propagation steps of the graph: A' is a subset of $A'(g)_{seq}$ when the sets are initialized (induction base). Expansion does not change A or $A(g)_{seq}$. Propagation does change both sets. So we assume, that A is a subset of $A(g)_{seq}$ and have to show that the inclusion $A' \subseteq A'(g)_{seq}$ holds (where $g \rightarrow_P g'$).

So let $\Gamma \in A'$. By assumption, Γ was added to A during the last step of propagation. So, Γ is in the greatest fixpoint of W^L, which means that $\exists(\Gamma, \Sigma) \in$

$L'_1.\forall(\Sigma, \Gamma') \in L'_2.\Gamma' \in A'$. *That is intuitively, either* Γ *itself was expanded since the last propagation or enough of the respective* Γ' *have been set to true since the last propagation.*

Assume, Γ *has been expanded. We show that* Γ *is also added to* $A(g)_{seq}$*, distinguishing several cases:*

- *The case that there is no premise to* Γ *is impossible since* Γ *is in the fixpoint.*
- *Any premises to* Γ *that are already known to be provable (i.e. all contained sequents are in A) or known to be not provable (i.e. one contained sequent is in E), are also contained in* $A(g)_{prem}$ *(or* $E(g)_{prem}$*).*
- *For any premises to* Γ *that have been encountered before but are still undecided, existence of a node that represents the premise is created.*
- *For any premises to* Γ *that have not been encountered before, a disjunct that represents the premise is created.*

For each premise Σ_j *to* Γ*, we distinguish upon the contained sequents:*

- *If* Σ_j *contains no sequents, the according node/disjunct is set to* $\top$ *s.t.* Γ *is added to* $A(g)_{seq}$ *upon propagation.*
- *Any already decided sequents* $\Gamma' \in A$ *(or* $\Gamma' \in E$*) from* Σ_j *are by assumption also contained in* $A(g)_{seq}$ *(or* $E(g)_{seq}$*).*
- *For any sequents in* Σ_j *that have been encountered before but are still undecided, existence of new node that represents the sequent is ensured.*
- *For any sequents in* Σ_j *that have not been encountered before, a conjunct that represents the sequent is created.*

Now, since we know that $\exists(\Gamma, \Sigma) \in L'_1.\forall(\Sigma, \Gamma') \in L'_2.\Gamma' \in A'$ *(i.e. we know that there is either an empty premise to* Γ *or a premise to* Γ' *whose sequents are all in* A'*. For each premise, we have a disjunct, representing it. For each contained sequent, we have conjunct, representing it. Since there is a premise to* Γ' *whose sequents are all in* A'*, by assumption we know that there is one disjunct whose conjuncts all simplify to* $\top$*.*

Circles?

For the other direction, we show that $A(g)_{seq}$ *is contained in* A*.*

Again we proceed by induction, initialization is the base case, expansion does not change the sets. If propagation adds a sequent Γ *to* $A(g)_{seq}$*, then one of the following three rules was applied, s.t.* Γ *had a representation in the simplified formula:*

1. $(\top \vee \phi) \rightarrow \top$ *(only one premise needs to be provable). Also set* $A(g')_{seq} = A(g)_{seq} \cup tops_{seq}$*, where* $tops_{seq}$ *are all the sequents that have a representation in* $\top$*. Also set* $A(g')_{prem} = A(g)_{prem} \cup tops_{prem}$*, where* $tops_{prem}$ *are all the sequents that have a representation in* $\top$*.*
2. $(\top \wedge \phi) \rightarrow \phi$ *(provable sequents are removed from the formula). Also set* $A(g')_{seq} = A(g)_{seq} \cup top$ *where top is the sequent that is represented by* $\top$*. Also set* $A(g')_{prem} = A(g)_{prem} \cup top$ *where top is the premise that is represented by* $\top$*.*

3. $(\psi^i = \top) \rightarrow (n_i \rightarrow \top)$. *Trivial proof formulas may be collapsed. Also set* $A(g')_{seq} = A(g)_{seq} \cup top$ *where top is the sequent that is represented by* $\top$. *Also set* $A(g')_{prem} = A(g)_{prem} \cup top$ *where top is the premise that is represented by* $\top$.

In the first case, Γ *was either represented by* $(\top \vee \phi)$ *or by* $\top$. *By induction, we only treat the first variant. We know that the disjunction was introduced due to the fact that there is at least one provable premise to* Γ *(the one having a representation in top). Obviously, this premise contains only provable sequents s.t.* $\Gamma \in A'$ *as required.*

In the second case, Γ *was represented by* $\top$. *How could it be represented by* $\top$*? Either because there just one empty premise to it or since there were other rules applied before (induction).*

In the third case, Γ *was represented* $\top$. *How could it be represented by* $\top$*? Either because there just one empty premise to it or since there were other rules applied before (induction).*

Intuitively, the fixpoint computation is just the same as constructing the complete graph formula every time and then simplifying it (but using previously known simplifications stored in A/E).

The new method gets rid of constructing the complete formula again and again by storing it and just *expanding* it with all the sequents which were added since the last propagation step.

In more detail, A equals the set of sequents which are represented by atoms in the (unsimplified but flattened) graph formula which may be set to $\top$ by a successful leaf or by simplification. Correspondingly, E equals the set of sequents which are represented by atoms in the (unsimplified but flattened) graph formula which may be set to $\bot$ by an unsuccessful leaf or by simplification.