Verified Decision Procedures for Modal Logics

Minchao Wu
Rajeev Goré

Research School of Computer Science
Australian National University

September 10, 2019

Overview

- What
  - Verified decision procedures for modal logics K, KT and S4
  - Verified backjumping for modal logic K
- How
  - Decision procedures as functions in Lean
  - With soundness, completeness and termination proved
- Literature
  - Tableaux with histories based on the sequent calculus given by Heuerding, Seyfried, and Zimmermann (HSZ)
  - Different proofs of correctness

Syntax

Definition (Syntax)

The syntax of formulas is given by the following grammar:

$$\mathbb{N} ::= 0 \mid S\mathbb{N}$$
$$\varphi ::= \mathbb{N} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \Box\varphi \mid \Diamond\varphi$$

We work with a simpler language NNF given by the following grammar:

$$\mathbb{N} ::= 0 \mid S\mathbb{N}$$
$$\varphi ::= \mathbb{N} \mid \neg\mathbb{N} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi$$

Semantics

Definition (Kripke models)

A Kripke model is a triple (S, R, V) where S is a set of states, and $R \subseteq S \times S$ and $V \subseteq \mathbb{N} \times S$ are two binary relations. A KT model is a Kripke model whose R is reflexive. An S4 model is a KT model whose R is transitive.

Definition (forcing)

Let $M = (S, R, V)$ be a Kripke model. The forcing relation $\Vdash$ is defined as follows:

$$
\begin{array}{ll}
(M, s) \Vdash n & \text{if } V(n, s) \\
(M, s) \Vdash \neg n & \text{if } (M, s) \nVdash n \\
(M, s) \Vdash \varphi \wedge \psi & \text{if } (M, s) \Vdash \varphi \;\text{ and }\; (M, s) \Vdash \psi \\
(M, s) \Vdash \varphi \vee \psi & \text{if } (M, s) \Vdash \varphi \;\text{ or }\; (M, s) \Vdash \psi \\
(M, s) \Vdash \Box\varphi & \text{if for all } t \in S, R(s, t) \text{ implies } (M, t) \Vdash \varphi \\
(M, s) \Vdash \Diamond\varphi & \text{if there exists } t \in S, R(s, t) \text{ and } (M, t) \Vdash \varphi
\end{array}
$$

Semantics

Definition (satisfiability)

Let M be a Kripke model. A state $s \in M$ satisfies a set Γ of formulas, written $(M, s) \vDash \Gamma$, if for all $\varphi \in \Gamma$, $(M, s) \Vdash \varphi$. A set Γ of formulas is satisfiable if there is a Kripke state that satisfies it. Otherwise, we say that Γ is unsatisfiable.

Calculus

Tableau for modal logic K:

$$(id)\ \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

$$(K)\ \frac{\Diamond\varphi, \Box\Sigma, \Gamma}{\varphi, \Sigma}$$

Tableau for modal logic KT and S4:

$$(T)\ \frac{\Box\varphi, \Gamma}{\varphi, \Box\varphi, \Gamma} \qquad (S4)\ \frac{\Diamond\varphi, \Box\Sigma, \Gamma}{\varphi, \Box\Sigma}$$

Alternative form

$$(K)\ \frac{\Diamond\varphi, \Box\Sigma, \Gamma}{\varphi, \Sigma}$$

should be understood as

$$(K)\ \frac{\Diamond\Delta, \Box\Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \dots \quad \varphi_n, \Sigma}$$

where

- $\Delta = \{\varphi_0 \dots \varphi_n\} \neq \varnothing$
- Γ is a set of literals
- Γ does not contain a pair $n, \neg n$

Strategy for K

$$(id)\ \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

$$(K)\ \frac{\Diamond\Delta, \Box\Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \ldots \quad \varphi_n, \Sigma}$$

Strategy:

- Start with the goal

Strategy for K

$$(id)\ \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

$$(K)\ \frac{\Diamond\Delta, \Box\Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \dots \quad \varphi_n, \Sigma}$$

Strategy:

- Start with the goal
- Call the decision procedure recursively on the lower sequents

Strategy for K

$$(id)\ \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

$$(K)\ \frac{\Diamond\Delta, \Box\Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \ldots \quad \varphi_n, \Sigma}$$

Strategy:

- Start with the goal
- Call the decision procedure recursively on the lower sequents
- Terminate if no rule is applicable, or a contradiction is found

Strategy for K

$$(id)\ \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

$$(K)\ \frac{\Diamond\Delta, \Box\Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \dots \quad \varphi_n, \Sigma}$$

Strategy:

- Start with the goal
- Call the decision procedure recursively on the lower sequents
- Terminate if no rule is applicable, or a contradiction is found
- Propagate the status upwards

Formalization

```
structure kripke (states : Type) :=
(val : ℕ → states → Prop)
(rel : states → states → Prop)

def sat {st} (k : kripke st) (s) (Γ : list nnf) :=
∀ φ ∈ Γ, force k s φ
```

- Rearranging `val` and `rel` during propagation is tedious.
- Defining a Kripke model from scratch whenever a rule is applied is also unsatisfying.

Uniform and cumulative models

- Tree models

```
inductive model
| cons : list ℕ → list model → model
```

- Interpretation functions

```
def mval : ℕ → model → bool
| p (cons v r) := p ∈ v

def mrel : model → model → bool
| (cons v r) m := m ∈ r
```

Uniform and cumulative models

- Builder

```
def builder : kripke model :=
{ val := λ n s, mval n s,
  rel := λ s₁ s₂, mrel s₁ s₂ }
```

Recall:

```
def sat {st} (k : kripke st) (s) (Γ : list nnf) :=
∀ φ ∈ Γ, force k s φ
```

Each state is a tree.

Handling the K-rule

$$(K)\ \frac{\Diamond\Delta, \Box\Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \dots \quad \varphi_n, \Sigma}$$

```
def tmap
{p : list nnf → Prop} (f : Π Γ, p Γ → node Γ):
Π Γ : list (list nnf), (∀ i∈Γ, p i) →
psum {i // i ∈ Γ ∧ unsatisfiable i}
     {x // batch_sat x Γ}
```

Example

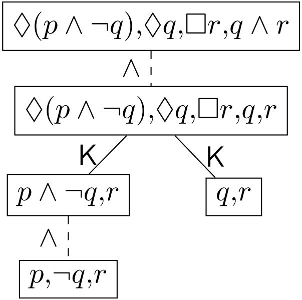

Figure: Expanding sequents ↓

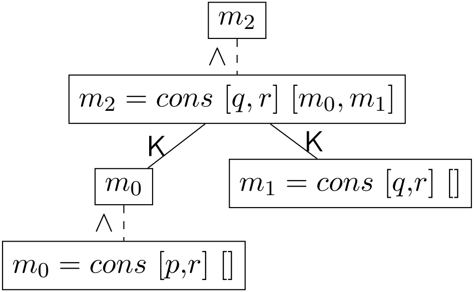

Figure: Constructing models ↑

Formalization

Return type:

```
inductive node (Γ : list nnf) : Type
| closed : unsatisfiable Γ → node
| open_  : {s // sat builder s Γ} → node
```

Decision procedure:

```
def tableau : Π Γ : list nnf, node Γ := ...
using_well_founded
{rel_tac := λ _ _, `[exact ⟨_, measure_wf node_size⟩]}
```

Wrapper:

```
def is_sat (Γ : list nnf) : bool := ...
```

Formalization

```
theorem correctness (Γ : list nnf) :
is_sat Γ = tt ↔
∃ (st : Type) (k : kripke st) s, sat k s Γ
```

Backjumping

Recall the $(\vee)$ rule:

$$(\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

- If the left child of the rule is unsatisfiable, there is a chance that the right child is also unsatisfiable.
- Happens when the principal formula $\varphi \vee \psi$ is *not responsible* for a contradiction.

Backjumping

A marking set M is recursively defined on closed branches.

Definition (responsibility)

1. For the id rule, $M = \{p, \neg p\}$.
2. Let M_l be the marking set of the lower sequent of the $\wedge$-rule.

$$M = \begin{cases} \{\varphi \wedge \psi\} \cup M_l & \text{if } \varphi \in M_l \text{ or } \psi \in M_l \\ M_l & \text{otherwise} \end{cases}$$

3. Let M_l and M_r be the marking sets of the left and right lower sequent of the $\vee$-rule respectively.

$$M = \begin{cases} \{\varphi \vee \psi\} \cup M_l \cup M_r & \text{if } \varphi \in M_l \text{ or } \psi \in M_r \\ M_l \cup M_r & \text{otherwise} \end{cases}$$

Backjumping

Definition (responsibility contd.)

Let l be the first unsatisfiable lower sequent of the K-rule, and M_l its marking set.

$$M = \Diamond(l.head) \cup \Box(l.tail \cap M_l)$$

Backjumping

$$(\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

Theorem (jumping)

If the left principal formula (i.e., φ) in the $(\vee)$ rule is not in the marking set of the left child, then the parent is unsatisfiable.

Theorem (marking property)

For each sequent φ, Γ, if φ is not in its marking set, then Γ is unsatisfiable.

Backjumping

Theorem (marking property revisited)

For each sequent Γ, if a subset $\Delta \subseteq \Gamma$ contains nothing in the marking set, then $\Gamma - \Delta$ is unsatisfiable.

Backjumping

```
def pmark (Γ m : list nnf) :=
∀ Δ, (∀ δ ∈ Δ, δ ∉ m) → Δ <+ Γ →
unsatisfiable (list.diff Γ Δ)
```

Force each closed node to carry a marking set with a proof of `pmark`.

```
inductive node (Γ : list nnf) : Type
| closed : Π m, unsatisfiable Γ → pmark Γ m → node
| open_  : {s // sat builder s Γ} → node
```

Performace

Subclass	K	K (backjumping)	FaCT++
branch_n	3	5	10
branch_p	1	3	10
d4_n	5	5	21
d4_p	6	7	21
dum_n	18	18	21
dum_p	9	17	21
grz_n	21	21	21
grz_p	6	7	21
lin_n	3	4	21
lin_p	6	7	21
path_n	10	10	21
path_p	2	12	21
ph_n	3	3	13
ph_p	2	3	7
poly_n	20	20	21
poly_p	19	21	21
t4p_n	7	7	21
t4p_p	7	12	21

Figure: Results on the LWB benchmark for K

Thoughts

- Decision procedures compute proofs.

Thoughts

- Decision procedures compute proofs.
- Proofs justify the programs.

Thoughts

- Decision procedures compute proofs.
- Proofs justify the programs.
- Dependent types help with correctness, termination and efficiency.

Thoughts

- Decision procedures compute proofs.
- Proofs justify the programs.
- Dependent types help with correctness, termination and efficiency.
- Different correctness proofs.

Thoughts

- Decision procedures compute proofs.
- Proofs justify the programs.
- Dependent types help with correctness, termination and efficiency.
- Different correctness proofs.
- Discovered two flaws in the original proofs.

KT issues

Non-termination:

$$(T)\ \frac{\Box\varphi, \Gamma}{\varphi, \Box\varphi, \Gamma}$$

Tableau with histories:

$$(id)\ \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\Sigma \mid \varphi \wedge \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\Sigma \mid \varphi \vee \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}$$

$$(T)\ \frac{\Sigma \mid \Box\varphi, \Gamma}{\Box\varphi, \Sigma \mid \varphi, \Gamma} \qquad (K)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Sigma}$$

Date structure

```
structure seqt : Type :=
(main : list nnf)
(hdld : list nnf)
...
```

Termination

Definition (modal degree)

Let Γ be a set of formulas. The degree of Γ is the maximal number of modal operators occurring in any formula $\varphi \in \Gamma$.

$$(id)\ \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\Sigma \mid \varphi \wedge \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\Sigma \mid \varphi \vee \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}$$

$$(T)\ \frac{\Sigma \mid \Box\varphi, \Gamma}{\Box\varphi, \Sigma \mid \varphi, \Gamma} \qquad (K)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Sigma}$$

For a sequent $\Sigma \mid \Gamma$, the pair $(degree(\Sigma \cup \Gamma), l(\Gamma))$ is decreasing under lexicographic order.

Strategy for KT

$$(id)\ \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\Sigma \mid \varphi \wedge \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\Sigma \mid \varphi \vee \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}$$

$$(T)\ \frac{\Sigma \mid \Box\varphi, \Gamma}{\Box\varphi, \Sigma \mid \varphi, \Gamma} \qquad (K)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Sigma}$$

Strategy:

- Start with the goal

Strategy for KT

$$(id)\ \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\Sigma \mid \varphi \wedge \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\Sigma \mid \varphi \vee \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}$$

$$(T)\ \frac{\Sigma \mid \Box\varphi, \Gamma}{\Box\varphi, \Sigma \mid \varphi, \Gamma} \qquad (K)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Sigma}$$

Strategy:

- Start with the goal
- Call the decision procedure recursively on the lower sequents

Strategy for KT

$$(id)\ \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\Sigma \mid \varphi \wedge \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\Sigma \mid \varphi \vee \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}$$

$$(T)\ \frac{\Sigma \mid \Box\varphi, \Gamma}{\Box\varphi, \Sigma \mid \varphi, \Gamma} \qquad (K)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Sigma}$$

Strategy:

- Start with the goal
- Call the decision procedure recursively on the lower sequents
- Terminate if no rule is applicable, or a contradiction is found

Strategy for KT

$$(id)\ \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\Sigma \mid \varphi \wedge \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\Sigma \mid \varphi \vee \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}$$

$$(T)\ \frac{\Sigma \mid \Box\varphi, \Gamma}{\Box\varphi, \Sigma \mid \varphi, \Gamma} \qquad (K)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Sigma}$$

Strategy:

- Start with the goal
- Call the decision procedure recursively on the lower sequents
- Terminate if no rule is applicable, or a contradiction is found
- Propagate the status upwards, but

Strategy for KT

$$(id)\ \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\Sigma \mid \varphi \wedge \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\Sigma \mid \varphi \vee \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}$$

$$(T)\ \frac{\Sigma \mid \Box\varphi, \Gamma}{\Box\varphi, \Sigma \mid \varphi, \Gamma} \qquad (K)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Sigma}$$

Strategy:

- Start with the goal
- Call the decision procedure recursively on the lower sequents
- Terminate if no rule is applicable, or a contradiction is found
- Propagate the status upwards, but
- Correctness is not obvious due to reflexivity

Correctness

Definition (reflexive sequents)

A sequent $\Sigma \mid \Gamma$ is called reflexive if for every $\Box\varphi \in \Sigma$, if a tree model $m := cons\ v\ l$ satisfies the following two conditions:

1. $m \vDash \Gamma$, and
2. for every $s \in l$, for every $\Box\psi \in \Sigma$, $s \Vdash \psi$.

then $m \Vdash \varphi$.

Theorem (KT sequents)

Let $\Sigma \mid \Gamma$ be a sequent generated by KT tableau. Then

1. *Σ contains only $\Box$-formulas.*
2. *$\Sigma \mid \Gamma$ is reflexive.*

Data structure

```
structure seqt : Type :=
(main : list nnf)
(hdld : list nnf)
-- reflexive sequents
(pmain : srefl main hdld)
-- there are only boxed formulas in hdld
(phdld : box_only hdld)
```

S4 issues

$$(id)\ \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \qquad (\wedge)\ \frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \qquad (\vee)\ \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}$$

$$(T)\ \frac{\Box\varphi, \Gamma}{\varphi, \Box\varphi, \Gamma} \qquad (S4)\ \frac{\Diamond\varphi, \Box\Sigma, \Gamma}{\varphi, \Box\Sigma}$$

- The measure trick $(degree(\Sigma \cup \Gamma), l(\Gamma))$ for KT does not work

$$(S4)\ \frac{\Box\Sigma \mid \Diamond\varphi, \Gamma}{\varnothing \mid \varphi, \Box\Sigma}$$

S4 tableau with histories

$$(id)\ \frac{A \mid\mid S \mid\mid H \mid\mid \Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}}$$

$$(\wedge)\ \frac{A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \varphi \wedge \psi, \Gamma}{A \mid\mid \varepsilon \mid\mid H \mid\mid \Sigma \mid \varphi, \psi, \Gamma}$$

$$(\vee)\ \frac{A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \varphi \vee \psi, \Gamma}{A \mid\mid \varepsilon \mid\mid H \mid\mid \Sigma \mid \varphi, \Gamma \quad A \mid\mid \varepsilon \mid\mid H \mid\mid \Sigma \mid \psi, \Gamma}$$

$$(\Box, \text{new})\ \frac{A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Box\varphi, \Gamma}{A \mid\mid \varepsilon \mid\mid \varnothing \mid\mid \Box\varphi, \Sigma \mid \varphi, \Gamma}\ (\Box\varphi \notin \Sigma)$$

$$(\Box, \text{dup})\ \frac{A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Box\varphi, \Gamma}{A \mid\mid \varepsilon \mid\mid H \mid\mid \Sigma \mid \varphi, \Gamma}\ (\Box\varphi \in \Sigma)$$

$$(S4)\ \frac{A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Diamond\varphi, \Gamma}{(\varphi, \Sigma), A \mid\mid (\varphi, \Sigma) \mid\mid \varphi, H \mid\mid \Sigma \mid \varphi, \Sigma}\ (\varphi \notin H)$$

Formalization

```
structure sseqt : Type :=
(goal : list nnf)
(a : list psig)
(s : sig) -- sig := option psig
(h b m: list nnf)
(ndh : list.nodup h)
(ndb : list.nodup b)
(sph : h <+~ closure goal)
(spb : b <+~ closure goal)
(sbm : m ⊆ closure goal)
(ha : ∀ φ ∈ h, (⟨φ, b⟩ : psig) ∈ a)
(hb : box_only b)
(ps₁ : Π (h : s ≠ none), dsig s h ∈ m)
(ps₂ : Π (h : s ≠ none), bsig s h ⊆ m)
```

Termination

Theorem (S4 termination)

Let $A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Gamma$ *be a sequent generated by S4 tableau and* $A' \mid\mid S' \mid\mid H' \mid\mid \Sigma' \mid \Gamma'$ *its root. The triple*

$$(l \circ cl(\Gamma') - l(\Sigma), l \circ cl(\Gamma') - l(H), l(\Gamma))$$

is decreasing under lexicographic order.

S4 ill-founded reasoning

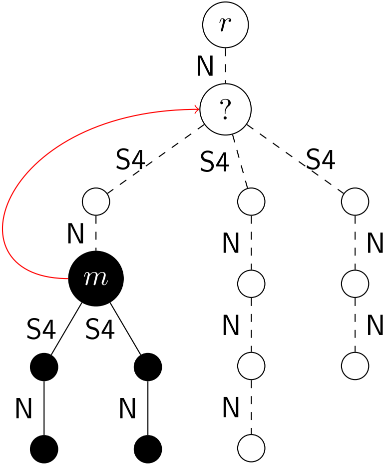

Figure: The red edge indicates that a loop-check is triggered at node m and a request is made. Black nodes are nodes with tree models constructed, and white nodes do not have a tree structure yet and their statuses are unknown to m. The node labeled r is the root.

S4 ill-founded reasoning

Difficulties:

- Need to know where the previous handling (S4-rule application) happened
- Cannot construct a tree model due to referring to nodes above
- Difficult to decide the status due to reffering to nodes with unexplored branches,
- in particular, the statuses of the referred nodes depend on the one being decided

Strategy for S4

- When no rule is applicable to $l = A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Gamma$ and Γ contains diamonds, a tree model m is constructed.

Strategy for S4

- When no rule is applicable to $l = A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Gamma$ and Γ contains diamonds, a tree model m is constructed.
- The tree model comes with some additional data, defined recursively in terms of upward propagation.

Strategy for S4

- When no rule is applicable to $l = A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Gamma$ and Γ contains diamonds, a tree model m is constructed.
- The tree model comes with some additional data, defined recursively in terms of upward propagation.
- The correctness of m is left open at the time it is constructed, instead, a set P of properties of m is proved.

Strategy for S4

- When no rule is applicable to $l = A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Gamma$ and Γ contains diamonds, a tree model m is constructed.
- The tree model comes with some additional data, defined recursively in terms of upward propagation.
- The correctness of m is left open at the time it is constructed, instead, a set P of properties of m is proved.
- P exploits the data contained in l and m, and is preserved by upward propagation. It is an invariant.

Strategy for S4

- When no rule is applicable to $l = A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Gamma$ and Γ contains diamonds, a tree model m is constructed.
- The tree model comes with some additional data, defined recursively in terms of upward propagation.
- The correctness of m is left open at the time it is constructed, instead, a set P of properties of m is proved.
- P exploits the data contained in l and m, and is preserved by upward propagation. It is an invariant.
- Propagate the tree model and the proofs of P upwards.

Strategy for S4

- When no rule is applicable to $l = A \mid\mid S \mid\mid H \mid\mid \Sigma \mid \Gamma$ and Γ contains diamonds, a tree model m is constructed.
- The tree model comes with some additional data, defined recursively in terms of upward propagation.
- The correctness of m is left open at the time it is constructed, instead, a set P of properties of m is proved.
- P exploits the data contained in l and m, and is preserved by upward propagation. It is an invariant.
- Propagate the tree model and the proofs of P upwards.
- Show that if the root sequent has a tree model m_r with P proved, then interpretation functions can be defined on a type induced by m_r to construct an S4 model m. It can be proved from P that $m \models \Gamma$.