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:

\[ \mathbb{N} ::= 0 \mid S\mathbb{N} \]
\[ \varphi ::= \mathbb{N} \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \varphi \rightarrow \varphi \mid \Box \varphi \mid \Diamond \varphi \]

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

\[ \mathbb{N} ::= 0 \mid S\mathbb{N} \]
\[ \varphi ::= \mathbb{N} \mid \neg \mathbb{N} \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \Box \varphi \mid \Diamond \varphi \]
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 \(\models\) is defined as follows:

\[
(M, s) \models n \quad \text{if} \quad V(n, s) \\
(M, s) \models \neg n \quad \text{if} \quad (M, s) \not\models n \\
(M, s) \models \varphi \land \psi \quad \text{if} \quad (M, s) \models \varphi \quad \text{and} \quad (M, s) \models \psi \\
(M, s) \models \varphi \lor \psi \quad \text{if} \quad (M, s) \models \varphi \quad \text{or} \quad (M, s) \models \psi \\
(M, s) \models \Box \varphi \quad \text{if} \quad \text{for all } t \in S, R(s, t) \text{ implies } (M, t) \models \varphi \\
(M, s) \models \Diamond \varphi \quad \text{if} \quad \text{there exists } t \in S, R(s, t) \text{ and } (M, t) \models \varphi
\]
Definition (satisfiability)

Let $M$ be a Kripke model. A state $s \in M$ satisfies a set $\Gamma$ of formulas, written $(M, s) \models \Gamma$, if for all $\varphi \in \Gamma$, $(M, s) \models \varphi$. A set $\Gamma$ of formulas is satisfiable if there is a Kripke state that satisfies it. Otherwise, we say that $\Gamma$ is unsatisfiable.
Tableau for modal logic K:

\[(id) \quad \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \quad (\land) \quad \frac{\varphi \land \psi, \Gamma}{\varphi, \psi, \Gamma} \quad (\lor) \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma} \quad (K) \quad \frac{\diamond \varphi, \square \Sigma, \Gamma}{\varphi, \Sigma}
\]

Tableau for modal logic KT and S4:

\[(T) \quad \frac{\square \varphi, \Gamma}{\varphi, \square \varphi, \Gamma} \quad (S4) \quad \frac{\diamond \varphi, \square \Sigma, \Gamma}{\varphi, \square \Sigma}
\]
Alternative form

\[(K) \quad \frac{\diamond \varphi, \Box \Sigma, \Gamma}{\varphi, \Sigma}\]

should be understood as

\[(K) \quad \frac{\diamond \Delta, \Box \Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \ldots \quad \varphi_n, \Sigma}\]

where

- \(\Delta = \{\varphi_0, \ldots, \varphi_n\} \neq \emptyset\)
- \(\Gamma\) is a set of literals
- \(\Gamma\) does not contain a pair \(n, \neg n\)
Strategy for K

\[
\begin{align*}
(id) & \quad \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \\
(\wedge) & \quad \frac{\varphi \land \psi, \Gamma}{\varphi, \psi, \Gamma} \\
(\vee) & \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}
\end{align*}
\]

\[
(K) \quad \frac{\Diamond \Delta, \Box \Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \cdots \quad \varphi_n, \Sigma}
\]

Strategy:

- Start with the goal
Strategy for $K$

\[
(id) \quad \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \quad \quad \quad (\wedge) \quad \frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \quad \quad \quad (\lor) \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}
\]

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

Strategy:

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

\[
\begin{align*}
(id) & \quad \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \\
(\land) & \quad \frac{\varphi \land \psi, \Gamma}{\varphi, \psi, \Gamma} \\
(\lor) & \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma} \\
(K) & \quad \frac{\lozenge \Delta, \square \Sigma, \Gamma}{\varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \ldots \quad \varphi_n, \Sigma}
\end{align*}
\]

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

\[
(id) \quad \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} \quad (\wedge) \quad \frac{\varphi \land \psi, \Gamma}{\varphi, \psi, \Gamma} \quad (\vee) \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma, \psi, \Gamma}
\]

(K) \quad \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
- Propagate the status upwards
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

\texttt{inductive model}

\begin{align*}
| \text{cons} : \text{list } \mathbb{N} & \rightarrow \text{list model } \rightarrow \text{model} \\
\end{align*}

▶ Interpretation functions

\texttt{def mval : } \mathbb{N} \rightarrow \text{model } \rightarrow \text{bool}

\begin{align*}
| \ p \ (\text{cons } v \ r) & := p \in v \\
\end{align*}

\texttt{def mrel : model } \rightarrow \text{model } \rightarrow \text{bool}

\begin{align*}
| \ (\text{cons } v \ r) \ m & := m \in r \\
\end{align*}
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

\[(K) \quad \varphi_0, \Sigma \quad \varphi_1, \Sigma \quad \ldots \quad \varphi_n, \Sigma \quad \Diamond \Delta, \Box \Sigma, \Gamma \]

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

\[ \Diamond (p \land \neg q), \Diamond q, \square r, q \land r \]

\[ \Diamond (p \land \neg q), \Diamond q, \square r, q, r \]

\[ p \land \neg q, r \]

\[ q, r \]

\[ p, \neg q, r \]

Figure: Expanding sequents ↓

\[ m_2 = \text{cons} [q, r] [m_0, m_1] \]

\[ m_0 = \text{cons} [p, r] [] \]

\[ m_1 = \text{cons} [q, r] [] \]

Figure: Constructing models ↑
Formalization

Return type:

\[
\text{inductive node } (\Gamma : \text{list nnf}) : \text{Type} \\
\quad \mid \text{closed} : \text{unsatisfiable } \Gamma \rightarrow \text{node} \\
\quad \mid \text{open} : \{s \ // \ \text{sat builder } s \ \Gamma\} \rightarrow \text{node}
\]

Decision procedure:

\[
\text{def tableau } : \Pi \ \Gamma : \text{list nnf}, \ \text{node } \Gamma := \ldots \\
\text{using_well_founded} \\
\{\text{rel_tac} := \lambda \_ \_, \ {\text{exact \langle \_\, , \ \text{measure_wf node_size}\rangle}}\}\]

Wrapper:

\[
\text{def is_sat } (\Gamma : \text{list nnf}) : \text{bool} := \ldots
\]
Formalization

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

Recall the (∨) rule:

\[
(\lor) \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma} \quad \frac{\varphi, \Gamma, \psi, \Gamma}{\psi, \Gamma}
\]

- 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 \( \varphi \lor \psi \) 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 $\land$-rule.

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

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

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

Definition (responsibility contd.)
Let \( l \) be the first unsatisfiable lower sequent of the K-rule, and \( M_l \) its marking set.

\[
M = \Diamond (l.\text{head}) \cup \Box (l.\text{tail} \cap M_l)
\]
Backjumping

\[(\lor) \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma \quad \psi, \Gamma}\]

Theorem (jumping)

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

Theorem (marking property)

For each sequent \(\varphi, \Gamma\), if \(\varphi\) is not in its marking set, then \(\Gamma\) is unsatisfiable.
Backjumping

Theorem (marking property revisited)

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

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

**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.
- Discovered two flaws in the original proofs.
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:

\[(T) \quad \square \varphi, \Gamma \quad \frac{\varphi, \square \varphi, \Gamma}{\square \varphi, \Gamma} \]

Tableau with histories:

\[(id) \quad \Sigma \mid n, \neg n, \Gamma \quad \frac{\text{unsatisfiable}}{\text{unsatisfiable}} \]

\[(\land) \quad \Sigma \mid \varphi \land \psi, \Gamma \quad \frac{\Sigma \mid \varphi, \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \quad \frac{\Sigma \mid \varphi \lor \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma} \]

\[(T) \quad \Sigma \mid \square \varphi, \Gamma \quad \frac{\square \varphi, \Sigma \mid \varphi, \Gamma}{\square \varphi, \Sigma \mid \varphi, \Gamma} \quad \frac{\Box \Sigma \mid \Diamond \varphi, \Gamma}{\emptyset \mid \varphi, \Sigma} \quad (K) \]

\[(K) \quad \frac{\Box \Sigma \mid \Diamond \varphi, \Gamma}{\emptyset \mid \varphi, \Sigma} \]
Date structure

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

Termination

Definition (modal degree)

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

\[
\begin{align*}
(id) & \quad \Sigma \mid n, \neg n, \Gamma & \quad \text{unsatisfiable} \\
(\land) & \quad \Sigma \mid \varphi \land \psi, \Gamma & \quad \Sigma \mid \varphi, \psi, \Gamma \\
(\lor) & \quad \Sigma \mid \varphi \lor \psi, \Gamma & \quad \Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma \\
(T) & \quad \Sigma \mid \Box \varphi, \Gamma & \quad \Box \varphi, \Sigma \mid \varphi, \Gamma \\
(K) & \quad \Box \Sigma \mid \Diamond \varphi, \Gamma & \quad \emptyset \mid \varphi, \Σ
\end{align*}
\]

For a sequent $\Sigma \mid \Gamma$, the pair $(\text{degree}(\Sigma \cup \Gamma), l(\Gamma))$ is decreasing under lexicographic order.
Strategy for KT

\[(id) \quad \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \quad (\land) \quad \frac{\Sigma \mid \varphi \land \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \quad (\lor) \quad \frac{\Sigma \mid \varphi \lor \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}\]

\[(T) \quad \frac{\Sigma \mid \square \varphi, \Gamma}{\square \varphi, \Sigma \mid \varphi, \Gamma} \quad (K) \quad \frac{\square \Sigma \mid \lozenge \varphi, \Gamma}{\emptyset \mid \varphi, \Sigma}\]

Strategy:

- Start with the goal
Strategy for KT

\[
(id) \quad \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \\
(\wedge) \quad \frac{\Sigma \mid \varphi \land \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \\
(\vee) \quad \frac{\Sigma \mid \varphi \lor \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma}
\]

\[
(T) \quad \frac{\Sigma \mid \Box \varphi, \Gamma}{\Box \varphi, \Sigma \mid \varphi, \Gamma} \\
(K) \quad \frac{\Box \Sigma \mid \Diamond \varphi, \Gamma}{\emptyset \mid \varphi, \Sigma}
\]

Strategy:

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

\[ \text{(id)} \quad \frac{\Sigma \mid n, \neg n, \Gamma}{\text{unsatisfiable}} \quad \text{(}\wedge\text{)} \quad \frac{\Sigma \mid \varphi \land \psi, \Gamma}{\Sigma \mid \varphi, \psi, \Gamma} \quad \text{(}\vee\text{)} \quad \frac{\Sigma \mid \varphi \lor \psi, \Gamma}{\Sigma \mid \varphi, \Gamma \quad \Sigma \mid \psi, \Gamma} \]

\[ \text{(T)} \quad \frac{\Sigma \mid \square \varphi, \Gamma}{\square \varphi, \Sigma \mid \varphi, \Gamma} \quad \text{(K)} \quad \frac{\square \Sigma \mid \lozenge \varphi, \Gamma}{\emptyset \mid \varphi, \Sigma} \]

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

\[(id) \quad \Sigma \vdash n, \neg n, \Gamma \quad \text{unsatisfiable}\]

\[(\wedge) \quad \Sigma \vdash \varphi \land \psi, \Gamma \quad \frac{\Sigma \vdash \varphi, \psi, \Gamma}{\Sigma \vdash \varphi, \Gamma \land \psi, \Gamma}\]

\[(\vee) \quad \Sigma \vdash \varphi \lor \psi, \Gamma \quad \frac{\Sigma \vdash \varphi, \Gamma}{\Sigma \vdash \varphi \lor \psi, \Gamma}\]

\[(T) \quad \Sigma \vdash \Box \varphi, \Gamma \quad \frac{\Sigma \vdash \varphi, \Gamma}{\Box \varphi, \Sigma \vdash \varphi, \Gamma}\]

\[(K) \quad \Box \Sigma \vdash \Diamond \varphi, \Gamma \quad \frac{\Box \Sigma \vdash \varphi, \Gamma}{\varnothing \vdash \varphi, \Sigma}\]

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

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

\[(T) \quad \frac{\Sigma \mid \Box \varphi, \Gamma}{\Box \varphi, \Sigma \mid \varphi, \Gamma} \quad (K) \quad \frac{\Box \Sigma \mid \Diamond \varphi, \Gamma}{\emptyset \mid \varphi, \Sigma} \]

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. $m \models \Gamma$, 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. $\Sigma$ 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

\[
\begin{align*}
(id) & \quad \frac{n, \neg n, \Gamma}{\text{unsatisfiable}} & (\wedge) & \quad \frac{\varphi \land \psi, \Gamma}{\varphi, \psi, \Gamma} & (\lor) & \quad \frac{\varphi \lor \psi, \Gamma}{\varphi, \Gamma \psi, \Gamma} \\
(T) & \quad \frac{\Box \varphi, \Gamma}{\varphi, \Box \varphi, \Gamma} & (S4) & \quad \frac{\Diamond \varphi, \Box \Sigma, \Gamma}{\varphi, \Box \Sigma}
\end{align*}
\]

▶ The measure trick \((\text{degree}(\Sigma \cup \Gamma), l(\Gamma))\) for KT does not work

\[
(S4) \quad \frac{\Box \Sigma | \Diamond \varphi, \Gamma}{\emptyset | \varphi, \Box \Sigma}
\]
S4 tableau with histories

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

\[(\land)\]
\[
\frac{A \parallel S \parallel H \parallel \Sigma \mid \varphi \land \psi, \Gamma}{A \parallel \varepsilon \parallel H \parallel \Sigma \mid \varphi, \psi, \Gamma}
\]

\[(\lor)\]
\[
\frac{A \parallel S \parallel H \parallel \Sigma \mid \varphi \lor \psi, \Gamma}{A \parallel \varepsilon \parallel H \parallel \Sigma \mid \psi, \Gamma}
\]

\[(\Box, \text{new})\]
\[
\frac{A \parallel S \parallel H \parallel \Sigma \mid \Box \varphi, \Gamma}{A \parallel \varepsilon \parallel \emptyset \parallel \Box \varphi, \Sigma \mid \varphi, \Gamma} \quad (\Box \varphi \notin \Sigma)
\]

\[(\Box, \text{dup})\]
\[
\frac{A \parallel S \parallel H \parallel \Sigma \mid \Box \varphi, \Gamma}{A \parallel \varepsilon \parallel H \parallel \Sigma \mid \varphi, \Gamma} \quad (\Box \varphi \in \Sigma)
\]

\[(S4)\]
\[
\frac{A \parallel S \parallel H \parallel \Sigma \mid \Diamond \varphi, \Gamma}{(\varphi, \Sigma), A \parallel (\varphi, \Sigma) \parallel \varphi, H \parallel \Sigma \mid \varphi, \Sigma} \quad (\varphi \notin H)
\]
Formalization

```lean
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 \( A \parallel S \parallel H \parallel \Sigma \parallel \Gamma \) be a sequent generated by S4 tableau and \( A' \parallel S' \parallel H' \parallel \Sigma' \parallel \Gamma' \) 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 referring 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 \( l = A \parallel S \parallel H \parallel \Sigma \parallel \Gamma \) and \( \Gamma \) contains diamonds, a tree model \( m \) is constructed.
Strategy for S4

- When no rule is applicable to \( l = A \parallel S \parallel H \parallel \Sigma \parallel \Gamma \) and \( \Gamma \) 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 \( l = A \parallel S \parallel H \parallel \Sigma | \Gamma \) and \( \Gamma \) 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 \( l = A \parallel S \parallel H \parallel \Sigma \parallel \Gamma \) and \( \Gamma \) 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 $l = A \parallel S \parallel H \parallel \Sigma \parallel \Gamma$ and $\Gamma$ 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.
When no rule is applicable to \( l = A \parallel S \parallel H \parallel \Sigma \parallel \Gamma \) and \( \Gamma \) 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 \( m \models \Gamma \).