Implementing and Evaluating Provers for First-order Modal Logics

Christoph Benzmüller\textsuperscript{1}, Jens Otten\textsuperscript{2} and Thomas Raths\textsuperscript{2}

\textsuperscript{1}Freie Universität Berlin
\textsuperscript{2}University of Potsdam
First-order Modal Logics (FMLs)

\[ p, q \ := \ P(t_1, \ldots, t_n) \ | \ (\neg p) \ | \ (p \lor q) \ | \ \Box p \ | \ (\forall x p) \]

are relevant for many applications, including

- planning
- natural language processing
- program verification
- modeling communication
- querying knowledge bases
- reasoning in expressive ontologies

Until recently, however, there has been

- a comparably large body of theory papers on FMLs
- but only one implemented prover! (GQML prover)
Our Contribution

Theory & implementation of new provers for FML:

- embedding into higher-order logic (LEO-II & Satallax)
- a connection calculus based prover (MleanCoP)
- a sequent calculus based prover (MleanSeP)
- a tableau based prover (MleanTAP)
- an instantiation based prover (f2p-MSPASS)

Moreover, we present

- a first comparative prover evaluation
- exploiting the new QMLTP library for FML
Our Contribution

Theory & implementation of new provers for FML:

- embedding into higher-order logic (LEO-II & Satallax)
- a connection calculus based prover (MleanCoP)
- a sequent calculus based prover (MleanSeP)
- a tableau based prover (MleanTAP)
- an instantiation based prover (f2p-MSPASS)

Moreover, we present

- a first comparative prover evaluation
- exploiting the new QMLTP library for FML
Experiment: 580 problems × 5 logics × 3 domain conditions × 6 provers × 600s tmo

8700 problems
Experiment: 580 problems × 5 logics × 3 domain conditions × 6 provers × 600s tmo

<table>
<thead>
<tr>
<th>Logic/Domain</th>
<th>f2p-MSPASS</th>
<th>MleanSeP</th>
<th>ATP system</th>
<th>LEO-II</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>v3.0</td>
<td>v1.2</td>
<td>v1.4.2</td>
<td>v2.2</td>
<td>v1.3</td>
<td>v1.2</td>
</tr>
<tr>
<td>K/varying</td>
<td>-</td>
<td>-</td>
<td>72</td>
<td>104</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/cumul.</td>
<td>70</td>
<td>121</td>
<td>89</td>
<td>122</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/constant</td>
<td>67</td>
<td>124</td>
<td>120</td>
<td>146</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D/varying</td>
<td>-</td>
<td>-</td>
<td>81</td>
<td>113</td>
<td>100</td>
<td>179</td>
</tr>
<tr>
<td>D/cumul.</td>
<td>79</td>
<td>130</td>
<td>100</td>
<td>133</td>
<td>120</td>
<td>200</td>
</tr>
<tr>
<td>D/constant</td>
<td>76</td>
<td>134</td>
<td>135</td>
<td>160</td>
<td>135</td>
<td>217</td>
</tr>
<tr>
<td>T/varying</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>170</td>
<td>138</td>
<td>224</td>
</tr>
<tr>
<td>T/cumul.</td>
<td>105</td>
<td>163</td>
<td>139</td>
<td>192</td>
<td>160</td>
<td>249</td>
</tr>
<tr>
<td>T/constant</td>
<td>95</td>
<td>166</td>
<td>173</td>
<td>213</td>
<td>175</td>
<td>269</td>
</tr>
<tr>
<td>S4/varying</td>
<td>-</td>
<td>-</td>
<td>140</td>
<td>207</td>
<td>169</td>
<td>274</td>
</tr>
<tr>
<td>S4/cumul.</td>
<td>121</td>
<td>197</td>
<td>166</td>
<td>238</td>
<td>205</td>
<td>338</td>
</tr>
<tr>
<td>S4/constant</td>
<td>111</td>
<td>197</td>
<td>200</td>
<td>261</td>
<td>220</td>
<td>352</td>
</tr>
<tr>
<td>S5/varying</td>
<td>-</td>
<td>-</td>
<td>169</td>
<td>248</td>
<td>219</td>
<td>359</td>
</tr>
<tr>
<td>S5/cumul.</td>
<td>140</td>
<td>-</td>
<td>215</td>
<td>297</td>
<td>272</td>
<td>438</td>
</tr>
<tr>
<td>S5/constant</td>
<td>131</td>
<td>-</td>
<td>237</td>
<td>305</td>
<td>272</td>
<td>438</td>
</tr>
</tbody>
</table>

Strongest Prover!
**Experiment:** 580 problems × 5 logics × 3 domain conditions × 6 provers × 600s tmo

<table>
<thead>
<tr>
<th>Logic/Domain</th>
<th>f2p-MSPASS v3.0</th>
<th>MleanSeP v1.2</th>
<th>ATP system</th>
<th>LEO-II v1.4.2</th>
<th>Satallax v2.2</th>
<th>MleanTAP v1.3</th>
<th>MleanCoP v1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>K/varying</td>
<td>-</td>
<td>-</td>
<td>72</td>
<td>104</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/cumul.</td>
<td>70</td>
<td>121</td>
<td>89</td>
<td>122</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/constant</td>
<td>67</td>
<td>124</td>
<td>120</td>
<td>146</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D/varying</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>81</td>
<td>113</td>
<td>100</td>
<td>179</td>
</tr>
<tr>
<td>D/cumul.</td>
<td>79</td>
<td>130</td>
<td>144100</td>
<td>133</td>
<td>120</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>D/constant</td>
<td>76</td>
<td>134</td>
<td>167135</td>
<td>160</td>
<td>135</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>T/varying</td>
<td>-</td>
<td>-</td>
<td>170120</td>
<td>170</td>
<td>138</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>T/cumul.</td>
<td>105</td>
<td>163</td>
<td>190139</td>
<td>192</td>
<td>160</td>
<td>249</td>
<td></td>
</tr>
<tr>
<td>T/constant</td>
<td>95</td>
<td>166</td>
<td>217173</td>
<td>213</td>
<td>175</td>
<td>269</td>
<td></td>
</tr>
<tr>
<td>S4/varying</td>
<td>-</td>
<td>-</td>
<td>140</td>
<td>207</td>
<td>169</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>S4/cumul.</td>
<td>121</td>
<td>197</td>
<td>218166</td>
<td>238</td>
<td>205</td>
<td>338</td>
<td></td>
</tr>
<tr>
<td>S4/constant</td>
<td>111</td>
<td>197</td>
<td>244200</td>
<td>261</td>
<td>220</td>
<td>352</td>
<td></td>
</tr>
<tr>
<td>S5/varying</td>
<td>-</td>
<td>-</td>
<td>169</td>
<td>248</td>
<td>219</td>
<td>359</td>
<td></td>
</tr>
<tr>
<td>S5/cumul.</td>
<td>140</td>
<td>-</td>
<td>215</td>
<td>297</td>
<td>272</td>
<td>438</td>
<td></td>
</tr>
<tr>
<td>S5/constant</td>
<td>131</td>
<td>-</td>
<td>237</td>
<td>305</td>
<td>272</td>
<td>438</td>
<td></td>
</tr>
</tbody>
</table>

**Strong improvement (≥ 25%)**
Experiment: 580 problems × 5 logics × 3 domain conditions × 6 provers × 600s tmo

<table>
<thead>
<tr>
<th>Logic/ Domain</th>
<th>f2p-MSPASS v3.0</th>
<th>MleanSeP v1.2</th>
<th>ATP system</th>
<th>LEO-II v1.4.2</th>
<th>Satallax v2.2</th>
<th>MleanTAP v1.3</th>
<th>MleanCoP v1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>K/varying</td>
<td>-</td>
<td>-</td>
<td>72</td>
<td>104</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/cumul.</td>
<td>70</td>
<td>121</td>
<td>89</td>
<td>122</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/constant</td>
<td>67</td>
<td>124</td>
<td>120</td>
<td>146</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D/varying</td>
<td>-</td>
<td>-</td>
<td>128 81</td>
<td>113</td>
<td>100</td>
<td>179</td>
<td>-</td>
</tr>
<tr>
<td>D/cumul.</td>
<td>79</td>
<td>130</td>
<td>144100</td>
<td>133</td>
<td>120</td>
<td>200</td>
<td>-</td>
</tr>
<tr>
<td>D/constant</td>
<td>76</td>
<td>134</td>
<td>167135</td>
<td>160</td>
<td>135</td>
<td>217</td>
<td>-</td>
</tr>
<tr>
<td>T/varying</td>
<td>-</td>
<td>-</td>
<td>170120</td>
<td>170</td>
<td>138</td>
<td>224</td>
<td>-</td>
</tr>
<tr>
<td>T/cumul.</td>
<td>105</td>
<td>163</td>
<td>190139</td>
<td>192</td>
<td>160</td>
<td>249</td>
<td>-</td>
</tr>
<tr>
<td>T/constant</td>
<td>95</td>
<td>166</td>
<td>217173</td>
<td>213</td>
<td>175</td>
<td>269</td>
<td>-</td>
</tr>
<tr>
<td>S4/varying</td>
<td>-</td>
<td>-</td>
<td>140</td>
<td>207</td>
<td>169</td>
<td>274</td>
<td>-</td>
</tr>
<tr>
<td>S4/cumul.</td>
<td>121</td>
<td>197</td>
<td>218166</td>
<td>238</td>
<td>205</td>
<td>338</td>
<td>-</td>
</tr>
<tr>
<td>S4/constant</td>
<td>111</td>
<td>197</td>
<td>244200</td>
<td>261</td>
<td>220</td>
<td>352</td>
<td>-</td>
</tr>
<tr>
<td>S5/varying</td>
<td>-</td>
<td>-</td>
<td>169</td>
<td>248</td>
<td>219</td>
<td>359</td>
<td>-</td>
</tr>
<tr>
<td>S5/cumul.</td>
<td>140</td>
<td>-</td>
<td>215</td>
<td>297</td>
<td>272</td>
<td>438</td>
<td>-</td>
</tr>
<tr>
<td>S5/constant</td>
<td>131</td>
<td>-</td>
<td>237</td>
<td>305</td>
<td>272</td>
<td>438</td>
<td>-</td>
</tr>
</tbody>
</table>

Results for 20 multimodal logic problems: LEO-II 15, Satallax 14
Embedding in HOL

Simple Types

\[ \alpha ::= \iota \mid \emptyset \mid \alpha_1 \rightarrow \alpha_2 \]
Embedding in HOL

Simple Types

$\alpha ::= \iota \mid \sigma \mid \alpha_1 \rightarrow \alpha_2$

Individuals

Booleans (True and False)

Functions/Predicates
Embedding in HOL

Simple Types
\[ \alpha ::= i | \mu | o | \alpha_1 \rightarrow \alpha_2 \]

Possible worlds

Individuals

Booleans (True and False)

Functions/Predicates
Embedding in HOL

\[ s, t \ ::= \ C_\alpha \mid x_\alpha \mid (\lambda x_\alpha \cdot s_\beta)_{\alpha \to \beta} \mid (s_{\alpha \to \beta} \ t_\alpha)_\beta \mid \\
(\neg_\alpha s_\alpha)_\alpha \mid (s_\alpha \lor_\alpha \ t_\alpha)_\alpha \mid (\forall x_\alpha \ t_\alpha)_\alpha \]

Constant Symbols
Variable Symbols
Embedding in HOL

\[ s, t \ ::= \ C_\alpha \mid x_\alpha \mid (\lambda x_\alpha \cdot s_\beta)_\alpha \rightarrow_\beta \mid (s_\alpha \rightarrow_\beta t_\alpha)_\beta \mid
\neg_\alpha \rightarrow_\alpha s_\alpha \alpha \mid (s_\alpha \vee_\alpha \rightarrow_\alpha \rightarrow_\alpha t_\alpha)_\alpha \mid (\forall x_\alpha t_\alpha)_\alpha \]

Constant Symbols
Variable Symbols
Abstraction
Application
Embedding in HOL

\[ s, t ::= C_\alpha \mid x_\alpha \mid (\lambda x_\alpha \cdot s_\beta)_{\alpha \to \beta} \mid (s_\alpha \to t_\alpha)_\beta \mid (\neg_{o \to o} s_0)_o \mid (s_0 \lor_{o \to o} t_0)_o \mid (\forall x_\alpha t_0)_o \]

Constant Symbols
Variable Symbols
Abstraction
Application
Logical Connectives
Embedding in HOL

\[
\text{HOL}
\]

\[
s, t \ ::= \ C_\alpha \mid x_\alpha \mid (\lambda x_\alpha \cdot s_\beta)_\alpha^{\rightarrow\beta} \mid (s_\alpha^{\rightarrow\beta} t_\alpha)_\beta \\
(\neg_{o^{\rightarrow o}} s_{o^{\rightarrow o}})_o \mid (s_{o^{\rightarrow o}} \lor_{o^{\rightarrow o}} t_{o^{\rightarrow o}})_o \mid (\forall x_\alpha t_o)_o
\]
Embedding in HOL

\[ \text{HOL} \quad s, t \quad ::= \quad C \mid x \mid (\lambda x.s) \mid (s \: t) \mid (\neg s) \mid (s \lor t) \mid (\forall x \: t) \]
Embedding in HOL

\[
\begin{align*}
\text{HOL} & \quad s, t ::= C \mid x \mid (\lambda x.s) \mid (s \; t) \mid (\neg s) \mid (s \lor t) \mid (\forall x t)
\end{align*}
\]

HOL (with Henkin semantics) is meanwhile very well understood.
Embedding in HOL

\[
\text{HOL} \quad s, t ::= C \mid x \mid (\lambda x.s) \mid (s \, t) \mid (\neg s) \mid (s \lor t) \mid (\forall x.t)
\]

HOL (with Henkin semantics) is meanwhile very well understood

HOL TPTP Infrastructure
Embedding in HOL

HOL $s, t ::= C \mid x \mid (\lambda x.s) \mid (s \ t) \mid (\neg s) \mid (s \lor t) \mid (\forall x.t)$

HOL (with Henkin semantics) is meanwhile very well understood

HOL TPTP Infrastructure

HOL Provers: LEO-II, Satallax, TPS, Isabelle, Nitpick, Refute
Embedding in HOL

HOL  
\[ s, t ::= C \mid x \mid (\lambda x.s) \mid (s \ t) \mid (\neg s) \mid (s \lor t) \mid (\forall x t) \]

FML  
\[ p, q ::= P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \lor q) \mid \Box p \mid (\forall x p) \]

\[ M, g, s \models \neg p \text{ iff not } M, g, s \models p \]
\[ M, g, s \models p \lor q \text{ iff } M, g, s \models p \text{ or } M, g, s \models q \]
\[ M, g, s \models \Box p \text{ iff } M, g, u \models p \text{ for all } u \text{ with } R(s, u) \]
\[ M, g, s \models \forall x p \text{ iff } M, [d/x]g, s \models p \text{ for all } d \in D \]
Embedding in HOL

\[ s, t ::= C \mid \mathsf{x} \mid (\lambda \mathsf{x}.s) \mid (s \ t) \mid (\neg s) \mid (s \lor t) \mid (\forall \mathsf{x} t) \]

\[ p, q ::= P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \lor q) \mid \Box p \mid (\forall \mathsf{x} p) \]

- \[ M, g, s \models \neg p \iff \text{not } M, g, s \models p \]
- \[ M, g, s \models p \lor q \iff M, g, s \models p \text{ or } M, g, s \models q \]
- \[ M, g, s \models \Box p \iff M, g, u \models p \text{ for all } u \text{ with } R(s, u) \]
- \[ M, g, s \models \forall \mathsf{x} p \iff M, [d/x]g, s \models p \text{ for all } d \in D \]

\textbf{FML in HOL:}

\[ \neg = \lambda p. \lambda w. \neg(pw) \]
\[ \lor = \lambda p. \lambda q. \lambda w. (pw) \lor (qw) \]
\[ \Box = \lambda p. \lambda w. \forall v (\neg (Rwv) \lor (pv)) \]
\[ \forall = \lambda h. \lambda w. \forall x (hxw) \]
Embedding in HOL

\[
\text{HOL} \quad s, t ::=} \quad C \mid x \mid (\lambda x. s) \mid (s \ t) \mid (\neg s) \mid (s \lor t) \mid (\forall x t)
\]

\[
\text{FML} \quad p, q ::=} \quad P(t_1, \ldots , t_n) \mid (\neg p) \mid (p \lor q) \mid \square p \mid (\forall x p)
\]

\[
M, g, s \models \neg p \quad \text{iff} \quad \text{not } M, g, s \models p
\]

\[
M, g, s \models p \lor q \quad \text{iff} \quad M, g, s \models p \text{ or } M, g, s \models q
\]

\[
M, g, s \models \square p \quad \text{iff} \quad M, g, u \models p \text{ for all } u \text{ with } R(s, u)
\]

\[
M, g, s \models \forall x p \quad \text{iff} \quad M, [d/x]g, s \models p \text{ for all } d \in D
\]

\[
\text{FML in HOL:} \quad \neg = \lambda p. \lambda w. \neg(pw)
\]

\[
\lor = \lambda p. \lambda q. \lambda w. (pw) \lor (qw)
\]

\[
\square = \lambda p. \lambda w. \forall v(\neg(Rwv) \lor (pv))
\]

\[
\forall = \lambda h. \lambda w. \forall x(hw)
\]

\[
\text{Meta-level notions:} \quad \text{valid} = \lambda p. \forall w. pw
\]
Embedding in HOL

HOL
\[ s, t ::= C \mid x \mid (\lambda x. s) \mid (s \ t) \mid (\neg s) \mid (s \lor t) \mid (\forall x t) \]

FML
\[ p, q ::= P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \lor q) \mid \Box p \mid (\forall x p) \]

\[ M, g, s \models \neg p \iff \text{not } M, g, s \models p \]
\[ M, g, s \models p \lor q \iff M, g, s \models p \text{ or } M, g, s \models q \]
\[ M, g, s \models \Box p \iff M, g, u \models p \text{ for all } u \text{ with } R(s, u) \]
\[ M, g, s \models \forall x p \iff M, [d/x]g, s \models p \text{ for all } d \in D \]

FML in HOL:
\[ \neg = \lambda p. \lambda w. \neg(pw) \]
\[ \lor = \lambda p. \lambda q. \lambda w. (pw) \lor (qw) \]
\[ \Box = \lambda p. \lambda w. \forall v (\neg(Rvw) \lor (pv)) \]
\[ \forall = \lambda h. \lambda w. \forall x (hxw) \]

Meta-level notions:  \[ \text{valid} = \lambda p. \forall w. pw \]

Soundness & Completeness
Embedding in HOL

\((\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)
Embedding in HOL

$$(\Diamond \exists x Pf x \land \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz$$

valid $$(\Diamond \exists x Pf x \land \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz$$
Embedding in HOL

\[(\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\]

valid \[(\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\]

\[= \lambda p. \lambda w. \forall v ( \neg (Rvw) \lor (pv))\]
Embedding in HOL

\[(\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\]

valid \[(\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\]

valid \[(\Diamond \exists x Pfx \land (\lambda w. \forall v (\neg (Rwv) \lor (\forall y (\Diamond Py \Rightarrow Qy) v)))) \Rightarrow \Diamond \exists z Qz\]

\[\Box = \lambda p. \lambda w. \forall v (\neg (Rwv) \lor (pv))\]
Embedding in HOL

\((\Diamond \exists x Pfx \land \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)

valid \((\Diamond \exists x Pfx \land \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)

valid \((\Diamond \exists x Pfx \land (\lambda w. \forall v (\neg (Rvw) \lor (\forall y (\Diamond Py \Rightarrow Qy) v)))) \Rightarrow \Diamond \exists z Qz\)

\[\forall w (\neg \neg (\neg \forall v (\neg Rvw) \lor \neg \forall x \neg P(fx)v) \lor \neg \forall v (\neg Rvw \lor \forall y (\neg \forall u (\neg Rvu \lor \neg Pyu) \lor Qyv))) \lor \neg \forall v (\neg Rvw \lor \neg \forall z \neg Qzv)\]
Embedding in HOL

\[ (\Diamond \exists x Pf \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz \]

valid \( (\Diamond \exists x Pf \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz \)

valid \( (\Diamond \exists x Pf \land (\lambda w. \forall v (\neg (Rwv) \lor (\forall y (\Diamond Py \Rightarrow Qy) v)))) \Rightarrow \Diamond \exists z Qz \)

\[ \forall w (\neg \neg (\neg \forall x \neg P(fx)v) \lor \neg \forall v (\neg Rwv \lor \forall y (\neg \forall u (\neg Rvu \lor \neg Pyu) \lor Qyv))) \lor \neg \forall v (\neg Rwv \lor \neg \forall z \neg Qzv) \]
Embedding in HOL

\[ (∗∃xPfx ∧ □∀y(∗Py ⇒ Qy)) ⇒ ∗∃zQz \]

valid \((∗∃xPfx ∧ □∀y(∗Py ⇒ Qy)) ⇒ ∗∃zQz \)

valid \((∗∃xPfx ∧ (λw.∀v(¬(Rvw) ∨ (∀y(∗Py⇒Qy)v)))) \) ⇒ ∗∃zQz \)

\[ ∀w(¬¬¬¬∀v(¬Rvw ∨ ¬¬∀x¬P(fx)v) ∨ ¬¬∀v(¬Rvw ∨ ∀y(¬¬∀u(¬Rvu∨¬Pyu)v)∨Qyv)) ∨ ¬¬∀v(¬Rvw ∨ ¬¬∀z¬Qzv)) \]

Axiomatization of properties of accessibility relation \(R\)

Logic K: no axioms

Logic T: \((\text{reflexive } R) \) — which expands into \(∀x Rxx\)

Logic S4: \((\text{reflexive } R) ∧ (\text{symmetric } R) ∧ (\text{transitive } R)\)

Logic ... ...
Embedding in HOL

\[ (∇∃xPfx \land (∀∀y(∇Py \implies Qy))) \implies (∇∃zQz) \]
valid \( (∇∃xPfx \land (∀∀y(∇Py \implies Qy))) \implies (∇∃zQz) \)
valid \( (∇∃xPfx \land (\lambda w. (∀v(¬(Rwv) \lor (∀y(∇Py \implies Qy) v)))) \implies (∇∃zQz) \)

\[ ∀w(¬¬¬¬∀v(¬Rwv \lor ¬¬∀x¬P(fx)v) \lor ¬∀v(¬Rwv \lor ∀y(¬¬∀u(¬Rvu \lor ¬Pyu) \lor Qyv))) \lor ¬∀v(¬Rwv \lor ¬¬∀z¬Qzv)) \]

Axiomatization of properties of accessibility relation \( R \)

Logic K: no axioms
Logic T: (reflexive \( R \)) — which expands into \( ∀x Rxx \)
Logic S4: (reflexive \( R \)) \land (symmetric \( R \)) \land (transitive \( R \))
Logic . . . . . .

This automates FML with constant domain semantics in HOL
Embedding in HOL

To obtain varying domain semantics:

- modify quantifier: \[ \forall = \lambda q \lambda w \forall x \text{ExistsInW} x w \Rightarrow qxw \]
- add non-emptiness axiom: \[ \forall w \exists x \text{ExistsInW} x w \]
- add designation axioms for constants \( c \): \[ \forall w \text{ExistsInW} c w \]
  (similar for function symbols)
Embedding in HOL

To obtain **varying domain semantics:**
- modify quantifier: \( \forall = \lambda q \lambda w \forall x \text{ExistsInW}xw \Rightarrow qxw \)
- add non-emptiness axiom: \( \forall w \exists x \text{ExistsInW}xw \)
- add designation axioms for constants \( c \): \( \forall w \text{ExistsInW}cw \)
  (similar for function symbols)

To obtain **cumulative domain semantics:**
- add axiom: \( \forall x \forall v \forall w \text{ExistsInW}xv \land Rvw \Rightarrow \text{ExistsInW}xw \)
Modal Sequent Calculus

- Extends the classical sequent calculus by modal rules for □ and ◊.
Modal Sequent Calculus

- Extends the classical sequent calculus by modal rules for $\square$ and $\Diamond$.
- E.g., for the modal logic T (cumulative domains) the modal rules are

\[
\Gamma, F \vdash \Delta \\
\Gamma, \square F \vdash \Delta \\
\square \text{-left}
\]

\[
\Gamma \vdash F, \Delta \\
\Gamma \vdash \Diamond F, \Delta \\
\Diamond \text{-right}
\]
Modal Sequent Calculus

- Extends the classical sequent calculus by modal rules for $\square$ and $\Diamond$.
- E.g., for the modal logic $\mathbf{T}$ (cumulative domains) the modal rules are

\[
\begin{align*}
\Gamma, F & \vdash \Delta \\
\Gamma, \square F & \vdash \Delta \quad \square\text{-left} \\
\Gamma, \square F & \vdash \Delta \\
\Gamma, \Diamond F & \vdash \Delta \quad \Diamond\text{-right} \\
\Gamma & \vdash \Delta \\
\Gamma, \Diamond F & \vdash \Delta \\
\Gamma, \square F & \vdash \Delta \quad \square\text{-right} \\
\Gamma, \Diamond F & \vdash \Delta \\
\Gamma(\square), F & \vdash \Delta(\Diamond) \quad \Diamond\text{-left}
\end{align*}
\]

with $\Gamma(\square) := \{ \square G \mid G \in \Gamma \}$ and $\Delta(\Diamond) := \{ \Diamond G \mid G \in \Delta \}$. 

Modal Sequent Calculus

- Extends the classical sequent calculus by modal rules for $\square$ and $\Diamond$.
- E.g., for the modal logic T (cumulative domains) the modal rules are

\[
\begin{align*}
\Gamma, F & \vdash \Delta & \Gamma, \square F & \vdash \Delta & \text{$\square$-left} \\
\Gamma, F & \vdash \Delta & \Gamma & \vdash \square F, \Delta & \text{$\square$-right} \\
\Gamma & \vdash F, \Delta & \Gamma & \vdash \Diamond F, \Delta & \Diamond$-right \\
\Gamma, F & \vdash \Delta(\Diamond) & \Gamma & \vdash \Diamond G, \Delta & \Diamond$-left
\end{align*}
\]

with $\Gamma(\square) := \{\square G \mid G \in \Gamma\}$ and $\Delta(\Diamond) := \{\Diamond G \mid G \in \Delta\}$.

- Similar modal rules for the modal logics K, K4, D, D4, S4, ... (but not for S5 or varying domain; for constant domain: add Barcan formulae).
Modal Sequent Calculus

- Extends the classical sequent calculus by **modal rules** for \( \Box \) and \( \Diamond \).
- E.g., for the modal logic T (cumulative domains) the **modal rules** are:

\[
\begin{align*}
\Gamma, F \vdash \Delta & \quad \Box - \text{left} \\
\Gamma, \Box F \vdash \Delta & \\
\Gamma \vdash F, \Delta & \quad \Diamond - \text{right} \\
\Gamma \vdash \Box F, \Delta & \\
\Gamma \vdash \Diamond F, \Delta & \quad \Diamond - \text{left} \\
\end{align*}
\]

with \( \Gamma(\Box) : = \{ \Box G \mid G \in \Gamma \} \) and \( \Delta(\Diamond) : = \{ \Diamond G \mid G \in \Delta \} \).

- **Similar modal rules** for the modal logics K, K4, D, D4, S4, ... (but not for S5 or varying domain; for constant domain: add **Barcan formulae**).
- **Analytic** (i.e. bottom-up) applications of some modal rules **delete formulae** from sequents, e.g., (for T) formulae in \( \Gamma \) and \( \Delta \) are deleted.
Modal Sequent Calculus – Example/Implementation

Example: \((\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)
Example: \((\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)

\[
\begin{align*}
\text{Pfd} \vdash Pfd, Qfd & \quad \text{axiom} \\
Pfd \vdash \Diamond Pfd, Qfd & \quad \Diamond\text{-right} \\
Pfd, \Diamond Pfd \Rightarrow Qfd \vdash Qfd & \quad \Rightarrow\text{-left} \\
Pfd, \Diamond Pfd \Rightarrow Qfd \vdash \exists z Qz & \quad \exists\text{-right} (z \setminus \text{fd}) \\
Pfd, \forall y (\Diamond Py \Rightarrow Qy) \vdash \exists z Qz & \quad \forall\text{-left} (y \setminus \text{fd}) \\
\exists x Pfx, \forall y (\Diamond Py \Rightarrow Qy) \vdash \exists z Qz & \quad \exists\text{-left} (x \setminus \text{d}) \\
\Diamond \exists x Pfx, \Box \forall y (\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z Qz & \quad \Diamond\text{-left} \\
\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z Qz & \quad \Box\text{-left} \\
\vdash (\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz & \quad \Rightarrow\text{-right}
\end{align*}
\]
Modal Sequent Calculus – Example/Implementation

Example: \((\Diamond \exists x Pfx \land \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)

\[
\begin{align*}
Pfd \vdash Pfd, Qfd & \quad \text{axiom} \\
\vdash \Diamond Pfd, Qfd & \quad \Diamond \text{-right} \\
\vdash Pfd, \Diamond Pfd \Rightarrow Qfd & \quad \Rightarrow \text{-left} \\
\vdash Pfd, Qfd \vdash Qfd & \quad  Pfd, Qfd \vdash Qfd \quad \exists \text{-right } \,(z \backslash fd) \\
\vdash Pfd, \forall y(\Diamond Py \Rightarrow Qy) & \vdash \exists z Qz \quad \forall \text{-left } \,(y \backslash fd) \\
\exists x Pfx, \forall y(\Diamond Py \Rightarrow Qy) & \vdash \exists z Qz \quad \exists \text{-left } \,(x \backslash d) \\
\vdash \Diamond \exists x Pfx, \Box \forall y(\Diamond Py \Rightarrow Qy) & \vdash \Diamond \exists z Qz \quad \Diamond \text{-left} \\
\vdash \exists x Pfx \land \Box \forall y(\Diamond Py \Rightarrow Qy) & \vdash \Diamond \exists z Qz \quad \land \text{-left} \\
\vdash (\Diamond \exists x Pfx \land \Box \forall y(\Diamond Py \Rightarrow Qy)) & \Rightarrow \Diamond \exists z Qz \quad \Rightarrow \text{-right}
\end{align*}
\]

**MleanSeP**: implementation of the modal sequent calculus in PROLOG.

- analytic proof search with free variables and a dynamic Skolemization.
Connections and Prefixes

Connection calculi use a connection-driven proof search, i.e. proof search is guided by identifying connections, which correspond to sequent axioms.
Connections and Prefixes

Connection calculi use a connection-driven proof search, i.e. proof search is guided by identifying connections, which correspond to sequent axioms.

- Connection is a pair of literals of the form \( \{P(s_1, .., s_n), \neg P(t_1, .., t_n)\} \).
- Connection corresponds to an axiom, if its literals unify under a first-order substitution \( \sigma_Q \), i.e. \( \sigma_Q(s_i) = \sigma_Q(t_i) \) for all \( 1 \leq i \leq n \).
Connections and Prefixes

Connection calculi use a connection-driven proof search, i.e. proof search is guided by identifying connections, which correspond to sequent axioms.

- **Connection** is a pair of literals of the form \( \{ P(s_1, \ldots, s_n), \neg P(t_1, \ldots, t_n) \} \).

- Connection corresponds to an axiom, if its literals unify under a first-order substitution \( \sigma_Q \), i.e. \( \sigma_Q(s_i) = \sigma_Q(t_i) \) for all \( 1 \leq i \leq n \).

To deal with modal logic a **prefix** \( p \) is assigned to each atomic formula \( A \).

- A **prefix** is a string over two alphabets of
  - \( \nu \): prefix variables (represent applications of \( \Box \)-left or \( \Diamond \)-right) and
  - \( \Pi \): prefix constants (represent applications of \( \Box \)-right or \( \Diamond \)-left).
Connections and Prefixes

Connection calculi use a connection-driven proof search, i.e. proof search is guided by identifying connections, which correspond to sequent axioms.

- **Connection** is a pair of literals of the form \( \{ P(s_1, \ldots, s_n), \neg P(t_1, \ldots, t_n) \} \).
- **Connection** corresponds to an axiom, if its literals unify under a first-order substitution \( \sigma_Q \), i.e. \( \sigma_Q(s_i) = \sigma_Q(t_i) \) for all \( 1 \leq i \leq n \).

To deal with modal logic a **prefix** \( p \) is assigned to each atomic formula \( A \).

- A **prefix** is a string over two alphabets of
  - \( \nu \): prefix variables (represent applications of \( \Box \)-left or \( \Diamond \)-right) and
  - \( \Pi \): prefix constants (represent applications of \( \Box \)-right or \( \Diamond \)-left).
- **Semantically**, a prefix denotes a specific world in a Kripke model.
Connections and Prefixes

Connection calculi use a connection-driven proof search, i.e. proof search is guided by identifying connections, which correspond to sequent axioms.

- **Connection** is a pair of literals of the form \{ P(s_1, .., s_n), \neg P(t_1, .., t_n) \}.

- Connection corresponds to an axiom, if its literals unify under a first-order substitution \( \sigma_Q \), i.e. \( \sigma_Q(s_i) = \sigma_Q(t_i) \) for all \( 1 \leq i \leq n \).

To deal with modal logic a **prefix** \( p \) is assigned to each atomic formula \( A \).

- **A prefix** is a string over two alphabets of
  - \( \nu \): prefix variables (represent applications of \( \Box \)-left or \( \Diamond \)-right)
  - \( \Pi \): prefix constants (represent applications of \( \Box \)-right or \( \Diamond \)-left).

- Semantically, a prefix denotes a specific world in a Kripke model.

The literals of a (modal) connection \( \{ A_1 : p_1, \neg A_2 : p_2 \} \) are not deleted by applications of modal (sequent rules) if their prefixes unify,

- i.e. \( \sigma_M(p_1) = \sigma_M(p_2) \) for a modal substituion \( \sigma_M : \nu \rightarrow (\nu \cup \Pi)^* \).
Connections and Prefixes – Example

Example 1: $\Diamond P \Rightarrow \Box P$ ("if possible $P$, then necessarily $P$")
Example 1: $\Diamond P \Rightarrow \Box P$ ("if possible $P$, then necessarily $P$")

- sequent calculus

  $\vdash P$  
  $\vdash \Diamond P \Rightarrow \Box P$  
  $\vdash \Diamond P \Rightarrow \Box P$  

$\Rightarrow$-right
3-Connection Calculus

Example 1: $\Diamond P \Rightarrow \Box P$ ("if possible $P$, then necessarily $P$")

- sequent calculus
  
  $\frac{P \vdash \Box P}{\Diamond P \vdash \Box P}$  $\Diamond$-left
  
  $\frac{P \Rightarrow \Box P}{\Diamond P \Rightarrow \Box P}$  $\Rightarrow$-right

- connection calculus
  
 connection $\{P : a, \neg P : b\}$

  $a \neq b \Rightarrow$ prefixes not unifiable

  $\Rightarrow \Rightarrow$ formula not valid
Connections and Prefixes – Example

Example 1: $\Diamond P \Rightarrow \Box P$ ("if possible $P$, then necessarily $P$")

- sequent calculus
  
  $\vdash P \Rightarrow ?$
  
  $\Diamond P \vdash \Box P$ $\Diamond$-left
  
  $\Diamond P \Rightarrow \Box P$ $\Rightarrow$-right

- connection calculus
  
  connection $\{P : a, \neg P : b\}$
  
  $a \neq b \Rightarrow$ prefixes not unifiable
  
  $\Rightarrow$ formula not valid

Example 2: $\Box Q \Rightarrow \Diamond Q$ ("if necessarily $Q$, then possibly $Q$")
Connections and Prefixes – Example

**Example 1:** $\Diamond P \Rightarrow \Box P$ (“if possible $P$, then necessarily $P$”)

- sequent calculus
  
  $P \vdash ?$
  
  $\Diamond P \vdash \Box P$  $\Diamond$-left
  
  $\Diamond P \Rightarrow \Box P$  $\Rightarrow$-right

- connection calculus
  
  connection $\{P : a, \neg P : b\}$
  
  $a \neq b \Rightarrow$ prefixes not unifiable
  
  $\Rightarrow$ formula not valid

**Example 2:** $\Box Q \Rightarrow \Diamond Q$ (“if necessarily $Q$, then possibly $Q$”)

- sequent calculus
  
  $Q \vdash Q$  \textit{axiom}
  
  $\Box Q \vdash \Diamond Q$  $\Box$-left
  
  $\Box Q \Rightarrow \Diamond Q$  $\Rightarrow$-right
Connections and Prefixes – Example

Example 1: ◊ P ⇒ □ P  ("if possible P, then necessarily P")

▶ sequent calculus

\[ \frac{P \vdash ?}{\diamond P \vdash \Box P} \quad \diamond\text{-left} \]
\[ \diamond P \Rightarrow \Box P \quad \Rightarrow\text{-right} \]

▶ connection calculus

connection \{P : a, \neg P : b\}
\[ a \neq b \rightsquigarrow \text{prefixes not unifiable} \]
\[ \implies \text{formula not valid} \]

Example 2: □ Q ⇒ ◊ Q  ("if necessarily Q, then possibly Q")

▶ sequent calculus

\[ Q \vdash Q \quad \text{axiom} \]
\[ \Box Q \vdash \diamond Q \quad \Box\text{-left} \]
\[ \Box Q \Rightarrow \diamond Q \quad \Rightarrow\text{-right} \]

▶ connection calculus

connection \{Q : V, \neg Q : W\}
\[ V = W \rightsquigarrow \sigma_M(V) = W \]
\[ \implies \text{formula valid} \]
Connections and Prefixes – Example

Example 1: $\Diamond P \Rightarrow \Box P$ (“if possible $P$, then necessarily $P$”)

- sequent calculus
  $\frac{P \vdash \top}{\Diamond P \vdash \Box P}$ $\Diamond$-left
  $\frac{\Diamond P \Rightarrow \Box P}{\Diamond P \Rightarrow \Box P}$ $\Rightarrow$-right

- connection calculus
  connection $\{P : a, \neg P : b\}$
  $a \neq b \not\Rightarrow$ prefixes not unifiable
  $\not\Rightarrow$ formula not valid

Example 2: $\Box Q \Rightarrow \Diamond Q$ (“if necessarily $Q$, then possibly $Q$”)

- sequent calculus
  $\frac{Q \vdash Q}{\Box Q \vdash \Diamond Q}$ axiom
  $\frac{\Box Q \vdash \Diamond Q}{\Box Q \vdash \Diamond Q}$ $\Box$-left
  $\frac{\Box Q \Rightarrow \Diamond Q}{\Box Q \Rightarrow \Diamond Q}$ $\Rightarrow$-right

- connection calculus
  connection $\{Q : V, \neg Q : W\}$
  $V = W \not\Rightarrow \sigma_M(V) = W$
  $\not\Rightarrow$ formula valid

Further restrictions on modal substitution $\sigma_M$ (and $\sigma_Q$):
- induced reduction ordering has to be irreflexive,
- accessibility condition determines specific modal logic ($D$, $T$, $S4$, ...),
- domain constraint determines specific domain condition (constant, ...).
A matrix is the (graphical) representation of a (first-order modal) formula used within the connection calculus.
A \textbf{matrix} is the (graphical) \textbf{representation} of a (first-order modal) formula used within the connection calculus.

- The \textbf{matrix} of a formula $F$ is a \textbf{set of clauses} that represent the disjunctive normal form of $F$ (or conjunctive normal form of $\neg F$).
- In the \textbf{prefixed matrix} of $F$ each literal is marked with its prefix.
Prefixed Matrix

A matrix is the (graphical) representation of a (first-order modal) formula used within the connection calculus.

- The matrix of a formula $F$ is a set of clauses that represent the disjunctive normal form of $F$ (or conjunctive normal form of $\neg F$).
- In the prefixed matrix of $F$ each literal is marked with its prefix.

**Example:** $(\Diamond \exists x \text{Pfx} \land \square \forall y (\Diamond \text{Py} \Rightarrow \text{Qy})) \Rightarrow \Diamond \exists z \text{Qz}$

- Prefixed matrix: $\{\{\neg \text{Pfd} : a_1\}, \{\text{Py} : V_1V_2, \neg \text{Qy} : V_1\}, \{\text{Qz} : V_3\}\}$
  
  ($x$ is an Eigenvariable, $y, z$ are free term variables, $a_1 \in \Pi$ is a prefix constant, $V_1, V_2, V_3 \in \nu$ are prefix variables; $d$ and $a_1$ are Skolem constants.)
A matrix is the (graphical) representation of a (first-order modal) formula used within the connection calculus.

- The matrix of a formula $F$ is a set of clauses that represent the disjunctive normal form of $F$ (or conjunctive normal form of $\neg F$).
- In the prefixed matrix of $F$ each literal is marked with its prefix.

**Example:** $\Diamond \exists x \text{Pfx} \land \Box \forall y (\Diamond P y \Rightarrow Q y) \Rightarrow \Diamond \exists z Q z$

- Prefixed matrix: $\{\{\neg P f d : a_1\}, \{P y : V_1 V_2, \neg Q y : V_1\}, \{Q z : V_3\}\}$

  ($x$ is a Eigenvariable, $y, z$ are free term variables, $a_1 \in \Pi$ is a prefix constant, $V_1, V_2, V_3 \in \nu$ are prefix variables; $d$ and $a_1$ are Skolem constants.)

- **Graphical representation:**

\[
\begin{bmatrix}
\begin{bmatrix}
\neg P f d : a_1
\end{bmatrix}
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix}
P y : V_1 V_2
\end{bmatrix}
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix}
\neg Q y : V_1
\end{bmatrix}
\begin{bmatrix}
Q z : V_3
\end{bmatrix}
\end{bmatrix}
\]
Modal Connection Calculus – Example/Implementation

Example: \[ (\Diamond \exists x \, Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z \, Qz \]
Example: \((\Diamond \exists x \, Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z \, Qz\)

\[
\left[
\begin{array}{c}
\neg Pfd : a_1 \\
Py : V_1 V_2 \\
\neg Qy : V_1
\end{array}
\right]
\Rightarrow
\left[
\begin{array}{c}
Qz : V_3
\end{array}
\right]
\]
Modal Connection Calculus – Example/Implementation

Example: \((\Box \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)

\[
\begin{bmatrix}
\neg Pfd : a_1 \\
Py : V_1 V_2 \\
\neg Qy : V_1 \\
Qz : V_3
\end{bmatrix}
\]

- with \(\sigma_Q(y)=fd\), \(\sigma_M(V_1)=a_1\), \(\sigma_M(V_2)=\varepsilon\)
Example: \((\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)

\[
\begin{array}{c}
\lnot Pfd : a_1 \\
Py : V_1 V_2 \\
\lnot Qy : V_1 \\
Qz : V_3
\end{array}
\]

- with \(\sigma_Q(y) = fd\), \(\sigma_Q(z) = fd\), \(\sigma_M(V_1) = a_1\), \(\sigma_M(V_2) = \varepsilon\), and \(\sigma_M(V_3) = a_1\)
Modal Connection Calculus – Example/Implementation

Example: \( (\Diamond \exists x Pfx \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz \)

\[
\begin{bmatrix}
\neg Pfd : a_1 \\
\neg Qy : V_1 \\
Qz : V_3
\end{bmatrix}
\]

- with \( \sigma_Q(y) = fd, \sigma_Q(z) = fd, \sigma_M(V_1) = a_1, \sigma_M(V_2) = \varepsilon, \) and \( \sigma_M(V_3) = a_1 \)

\( \Rightarrow \) formula is valid for T/S4 (constant/cumulative/varying domains).
Example: \((\Box \exists x Pfx \land \Box \forall y (\Box Py \Rightarrow Qy)) \Rightarrow \Box \exists z Qz\)

\[
\begin{bmatrix}
\neg Pfd : a_1 \\
\end{bmatrix}
\begin{bmatrix}
P y : V_1 V_2 \\
\neg Qy : V_1 \\
Qz : V_3 \\
\end{bmatrix}
\]

- with \(\sigma_Q(y)=fd\), \(\sigma_Q(z)=fd\), \(\sigma_M(V_1)=a_1\), \(\sigma_M(V_2)=\epsilon\), and \(\sigma_M(V_3)=a_1\)

\(\Rightarrow\) formula is valid for T/S4 (constant/cumulative/varying domains).

**MleanCoP**: very compact implementation of modal connection calculus.

- based on **leanCoP**, a compact PROLOG prover for classical logic.
Modal Connection Calculus – Example/Implementation

Example: \((\Diamond \exists x \ Pfx \land \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\)

- \([\neg Pfd : a_1]\)
- \([Py : V_1 V_2]\)
- \([\neg Qy : V_1]\)
- \([Qz : V_3]\)

- with \(\sigma_Q(y)=fd, \sigma_Q(z)=fd, \sigma_M(V_1)=a_1, \sigma_M(V_2)=\varepsilon,\) and \(\sigma_M(V_3)=a_1\)

\(\implies\) formula is valid for T/S4 (constant/cumulative/varying domains).

MleanCoP: very compact implementation of modal connection calculus.

- based on leanCoP, a compact PROLOG prover for classical logic.
- 1. MleanCoP performs a classical proof search and collects prefixes.
  2. prefixes are unified using a special prefix unification algorithm.
Modal Connection Calculus – Example/Implementation

Example: \( (◊ \exists x Pfx \land \Box \forall y (◊ Py \Rightarrow Qy)) \Rightarrow ◊ \exists z Qz \)

\[
\begin{bmatrix}
\neg Pfd : a_1 \\
\neg Qy : V_1 \\
P y : V_1 V_2 \\
Qz : V_3
\end{bmatrix}
\]

- with \( \sigma_Q(y)=fd, \sigma_Q(z)=fd, \sigma_M(V_1)=a_1, \sigma_M(V_2)=\varepsilon, \) and \( \sigma_M(V_3)=a_1 \)

\( \Rightarrow \) formula is valid for T/S4 (constant/cumulative/varying domains).

MleanCoP: very compact implementation of modal connection calculus.

- based on leanCoP, a compact PROLOG prover for classical logic.

- 1. MleanCoP performs a classical proof search and collects prefixes.
- 2. Prefixes are unified using a special prefix unification algorithm.

- additional techniques: regularity, lemmata, restricted backtracking, ...
Modal Connection Calculus – Example/Implementation

Example:  $(\Box \exists x Pfx \land \Box \forall y(\Box Py \Rightarrow Qy)) \Rightarrow \Box \exists z Qz$

- with $\sigma_Q(y)=f$, $\sigma_Q(z)=f$, $\sigma_M(V_1)=a_1$, $\sigma_M(V_2)=\varepsilon$, and $\sigma_M(V_3)=a_1$
- $\implies$ formula is valid for T/S4 (constant/cumulative/varying domains).

**MleanCoP**: very compact implementation of modal connection calculus.

- based on leanCoP, a compact PROLOG prover for classical logic.
- 1. MleanCoP performs a classical proof search and collects prefixes.
   2. prefixes are unified using a special prefix unification algorithm.
- additional techniques: regularity, lemmata, restricted backtracking, ...
- available at [http://www.leancop.de](http://www.leancop.de) (GNU GPL license).
The source code of the leanCoP core prover for first-order classical logic.

```prolog
prove([],_,_,_,_).
prove([Lit |Cla],Path,PathLim,Lem, Set) :-
  (member(LitC,[Lit |Cla]), member(LitP,Path), LitC==LitP),
  (-NegLit=Lit;-Lit=NegLit) ->
  ( member(LitL,Lem), Lit ==LitL
    ;
    member(NegL ,Path), unify_with_occurs_check(NegL,NegLit)
    ;
    lit(NegLit , Cla1,Grnd1),
    ( Grnd1=g -> true ; length(Path,K), K<PathLim -> true ;
      pathlim -> assert(pathlim), fail ),
    prove(Cla1,[Lit |Path],PathLim,Lem, Set)
    ),
  ( member(cut,Set) -> ! ; true ),
prove(Cla,Path,PathLim,[Lit |Lem], Set).
```

C. Benzmüller / J. Otten / T. Raths (Freie Universität Berlin, University of Potsdam)
MleanCoP – Source Code

The source code of the MleanCoP core prover for first-order modal logic.

(1)    prove([],_,_,_,[[]],[]),
(2)    prove([Lit:Pre|Cla],Path,PathLim,Lem,[PreSet,FreeV],Set) :-
(3)        \+ (member(LitC,[Lit:Pre|Cla]), member(LitP,Path), LitC==LitP),
(4)        (-NegLit=Lit;-Lit=NegLit) ->
(5)            ( member(LitL,Lem), Lit:Pre==LitL, PreSet3=[], FreeV3=[]
(6)        ;
(7)        member(NegL:PreN,Path), unify_with_occurs_check(NegL,NegLit),
(8)        \+ \+ prefix_unify([Pre=PreN]), PreSet3=[Pre=PreN], FreeV3=[]
(9)        ;
(10)       lit(NegLit:PreN,FV:Cla1,Grnd1),
(11)       \+ \+ prefix_unify([Pre=PreN]),
(12)       ( Grnd1=g -> true ; length(Path,K), K<PathLim -> true ;
(13)          \+ pathlim -> assert(pathlim), fail ),
(14)       prove(Cla1,[Lit:Pre|Path],PathLim,Lem,[PreSet1,FreeV1],Set),
(15)       PreSet3=[Pre=PreN|PreSet1], append(FreeV1,FV,FreeV3)
(16)       ),
(17)       ( member(cut,Set) -> ! ; true ),
(18)       prove(Cla,Path,PathLim,[Lit:Pre|Lem],[PreSet2,FreeV2],Set),
(19)       append(PreSet3,PreSet2,PreSet), append(FreeV2,FreeV3,FreeV).
Tableau Calculus

Extends the classical tableau calculus by adding modal rules for □ and ◊ and a prefix to every formula in the tableau.
Tableau Calculus

Extends the classical tableau calculus by adding modal rules for $\square$ and $\Diamond$ and a prefix to every formula in the tableau.

- Branch is closed iff it contains a connection $\{A_1 : p_1, \neg A_2 : p_2\}$ with $\sigma_Q(A_1) = \sigma_Q(A_2)$ and $\sigma_M(p_1) = \sigma_M(p_2)$ for substitutions $\sigma_Q / \sigma_M$. 
Tableau Calculus

Extends the classical tableau calculus by adding modal rules for $\square$ and $\Diamond$ and a prefix to every formula in the tableau.

- Branch is closed iff it contains a connection $\{A_1 : p_1, \neg A_2 : p_2\}$ with $\sigma_Q(A_1) = \sigma_Q(A_2)$ and $\sigma_M(p_1) = \sigma_M(p_2)$ for substitutions $\sigma_Q/\sigma_M$.

- Similar to connection calculus, but proof search not connection-driven.
Tableau Calculus

Extends the classical tableau calculus by adding modal rules for $\Box$ and $\Diamond$ and a prefix to every formula in the tableau.

- Branch is closed iff it contains a connection $\{A_1 : p_1, \neg A_2 : p_2\}$ with $\sigma_Q(A_1) = \sigma_Q(A_2)$ and $\sigma_M(p_1) = \sigma_M(p_2)$ for substitutions $\sigma_Q/\sigma_M$.
- Similar to connection calculus, but proof search not connection-driven.
- Particular logic specified by constraints on the modal substitution $\sigma_M$. 
Tableau Calculus

Extends the classical tableau calculus by adding modal rules for $\Box$ and $\diamond$ and a prefix to every formula in the tableau.

- Branch is closed iff it contains a connection $\{A_1 : p_1, \neg A_2 : p_2\}$ with $\sigma_Q(A_1) = \sigma_Q(A_2)$ and $\sigma_M(p_1) = \sigma_M(p_2)$ for substitutions $\sigma_Q/\sigma_M$.

- Similar to connection calculus, but proof search not connection-driven.

- Particular logic specified by constraints on the modal substitution $\sigma_M$.

MleanTAP: compact implementation of the modal tableau calculus.

- based on ileanTAP, a compact PROLOG prover for intuitionistic logic.
Tableau Calculus

Extends the classical tableau calculus by adding modal rules for □ and ◇ and a prefix to every formula in the tableau.

- Branch is closed iff it contains a connection \{A_1 : p_1, \neg A_2 : p_2\} with \(\sigma_Q(A_1) = \sigma_Q(A_2)\) and \(\sigma_M(p_1) = \sigma_M(p_2)\) for substitutions \(\sigma_Q/\sigma_M\).

- Similar to connection calculus, but proof search not connection-driven.

- Particular logic specified by constraints on the modal substitution \(\sigma_M\).

MleanTAP: compact implementation of the modal tableau calculus.

- based on ileanTAP, a compact PROLOG prover for intuitionistic logic.

- 1. MleanTAP performs a classical proof search and collects prefixes.
  2. prefixes are unified using a special prefix unification algorithm.
Tableau Calculus

Extends the classical tableau calculus by adding modal rules for $\Box$ and $\Diamond$ and a prefix to every formula in the tableau.

- Branch is closed iff it contains a connection \( \{A_1 : p_1, \neg A_2 : p_2\} \) with \( \sigma_Q(A_1) = \sigma_Q(A_2) \) and \( \sigma_M(p_1) = \sigma_M(p_2) \) for substitutions \( \sigma_Q/\sigma_M \).
- Similar to connection calculus, but proof search not connection-driven.
- Particular logic specified by constraints on the modal substitution \( \sigma_M \).

MleanTAP: compact implementation of the modal tableau calculus.

- based on ileanTAP, a compact PROLOG prover for intuitionistic logic.
- 1. MleanTAP performs a classical proof search and collects prefixes.
  2. prefixes are unified using a special prefix unification algorithm.
- available at http://www.leancop.de/mleantap/ (GPL license).
Instance-Based Method

1. step: generate and add formula instances to the formula and ground it (remove quantifiers, replace variables by a single constant).

2. step: use propositional modal prover to find proof or countermodel; if no proof is found, go to first step and generate more instances.
Instance-Based Method

1. step: generate and add formula instances to the formula and ground it (remove quantifiers, replace variables by a single constant).

2. step: use propositional modal prover to find proof or countermodel; if no proof is found, go to first step and generate more instances.

Example: \((\lozenge Pfd \land \Box \forall y (\lozenge Py \Rightarrow Qy)) \Rightarrow \lozenge \exists z Qz\).
Instance-Based Method

1. step: generate and add formula instances to the formula and ground it (remove quantifiers, replace variables by a single constant).

2. step: use propositional modal prover to find proof or countermodel; if no proof is found, go to first step and generate more instances.

Example: \((\Box P d \land \square \forall y (\Box P y \Rightarrow Q y)) \Rightarrow \Box \exists z Q z\).

first instance is not valid: \((\Box P d \land \square (\Box P a \Rightarrow Q a)) \Rightarrow \Box Q a\)
**Instance-Based Method**

1. **step**: generate and **add formula instances** to the formula and **ground** it (remove quantifiers, replace variables by a single constant).

2. **step**: use **propositional modal prover** to find proof or countermodel; if no proof is found, go to first step and generate more instances.

**Example**: \((\Diamond Pfd \land \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz\).

- **first instance** is not valid: \((\Diamond Pfd \land \Box (\Diamond Pa \Rightarrow Qa)) \Rightarrow \Diamond Qa\)
- **second instance** is valid:

\[(\Diamond Pfd \land \Box ((\Diamond Pa \Rightarrow Qa) \land (\Diamond Pfd \Rightarrow Qfd))) \Rightarrow \Diamond (Qa \lor Qfd)\]
Instance-Based Method

1. step: generate and add formula instances to the formula and ground it (remove quantifiers, replace variables by a single constant).

2. step: use propositional modal prover to find proof or countermodel; if no proof is found, go to first step and generate more instances.

Example: \( (\Box Pfd \land \square \forall y (\Box Py \Rightarrow Qy)) \Rightarrow \Box \exists z Qz \).

- first instance is not valid: \( (\Box Pfd \land \Box (\Box Pa \Rightarrow Qa)) \Rightarrow \Box Qa \)
- second instance is valid:

\[
(\Box Pfd \land \Box ((\Box Pa \Rightarrow Qa) \land (\Box Pfd \Rightarrow Qfd))) \Rightarrow \Box (Qa \lor Qfd)
\]

f2p-MSPASS: instance-based prover for first-order modal logic.

- first component first2p, adds and grounds non-clauses instances.
- propositional modal prover MSPASS is used to find proofs.
- works for formula containing only universal/only existential quantifiers.
The QMLTP Problem Library

- The **Quantified Modal Logic Theorem Proving** problem library ... is available at [http://www.iltp.de/qmltp](http://www.iltp.de/qmltp).

- **Purpose**: put **evaluation** of modal provers onto a firm basis and stimulate the development of more **efficient** modal provers.
The QMLTP Problem Library

- The Quantified Modal Logic Theorem Proving problem library ...
  is available at http://www.iltp.de/qmltp.

- **Purpose**: put **evaluation** of modal provers onto a firm basis and stimulate the development of more **efficient** modal provers.

- QMLTP library v1.1: **600 problems** divided into **11 problem classes**.

- Header of each problem file includes, e.g., description, **difficulty rating** (0=easy to 1.0=difficult), **status** (Theorem/Non-Theorem/Unsolved).

- Status and rating information provided for the modal logics **K**, **D**, **T**, **S4**, and **S5** with **constant**, **cumulative** or **varying** domains.
The QMLTP Problem Library

- The Quantified Modal Logic Theorem Proving problem library ... is available at http://www.iltp.de/qmltp.

- Purpose: put evaluation of modal provers onto a firm basis and stimulate the development of more efficient modal provers.

- QMLTP library v1.1: 600 problems divided into 11 problem classes.

- Header of each problem file includes, e.g., description, difficulty rating (0=easy to 1.0=difficult), status (Theorem/Non-Theorem/Unsolved).

- Status and rating information provided for the modal logics $K$, $D$, $T$, $S4$, and $S5$ with constant, cumulative or varying domains.

- TPTP syntax (for classical logic) is extended by the modal operators $\#box$ and $\#dia$ representing □ and ◇, respectively.
QMLTP Library – Problem Sample

% File : SYM001+1 : QMLTP v1.1
% Domain : Syntactic (modal)
% Problem : Barcan scheme instance. (Ted Sider’s qml wwf 1)
% Version : Especial.
% English : if for all x necessarily f(x), then it is necessary that for all x f(x)
% Source : [Sid09]
% Names : instance of the Barcan formula
% Status : varying cumulative constant
% K Non-Theorem Non-Theorem Theorem v1.1
% D Non-Theorem Non-Theorem Theorem v1.1
% T Non-Theorem Non-Theorem Theorem v1.1
% S4 Non-Theorem Non-Theorem Theorem v1.1
% S5 Non-Theorem Theorem Theorem v1.1
% Rating : varying cumulative constant
% K 0.50 0.75 0.25 v1.1
% D 0.75 0.83 0.17 v1.1
% T 0.50 0.67 0.17 v1.1
% S4 0.50 0.67 0.17 v1.1
% S5 0.50 0.20 0.20 v1.1

qmf(con,conjecture,
(( ! [X] : (#box : ( f(X) ) ) ) => (#box : ( ! [X] : ( f(X) ) )))).
Conclusion

Summary:

- overview of 5 sound FML provers in one talk (!)
- used QMLTP library for first evaluation
- one older system excluded because of soundness issues
- strongest provers: MleanCoP followed by Satallax
- best coverage: HOL approach
Conclusion

Summary:

- overview of 5 sound FML provers in one talk (!)
- used QMLTP library for first evaluation
- one older system excluded because of soundness issues
- strongest provers: MleanCoP followed by Satallax
- best coverage: HOL approach

Future work includes:

- extension of calculi/implementations to further modal logics
- improvements of the presented provers
- extensions of the QMLTP library and related infrastructure
Thank you!

Any questions?
## Experiments using the QMLTP Library

<table>
<thead>
<tr>
<th>Logic/Domain</th>
<th>ATP system</th>
<th>Domain</th>
<th>f2p-MSPASS</th>
<th>MleanSeP</th>
<th>LEO-II</th>
<th>Satallax</th>
<th>MleanTAP</th>
<th>MleanCoP</th>
</tr>
</thead>
<tbody>
<tr>
<td>K/varying</td>
<td>0/529</td>
<td>-</td>
<td>165/356</td>
<td>-</td>
<td>0/511</td>
<td>50/349</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/cumul.</td>
<td>88/363</td>
<td>4/471</td>
<td>0/511</td>
<td>50/349</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K/constant</td>
<td>42/405</td>
<td>2/471</td>
<td>12/481</td>
<td>45/328</td>
<td>0/492</td>
<td>293/173</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D/varying</td>
<td>0/519</td>
<td>-</td>
<td>0/477</td>
<td>-</td>
<td>0/492</td>
<td>293/173</td>
<td>0/472</td>
<td>194/171</td>
</tr>
<tr>
<td>D/cumul.</td>
<td>33/407</td>
<td>0/461</td>
<td>0/500</td>
<td>0/464</td>
<td>0/472</td>
<td>194/171</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D/constant</td>
<td>33/411</td>
<td>0/462</td>
<td>2/466</td>
<td>0/425</td>
<td>0/456</td>
<td>167/169</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T/varying</td>
<td>0/478</td>
<td>-</td>
<td>30/320</td>
<td>-</td>
<td>0/493</td>
<td>121/223</td>
<td>0/472</td>
<td>76/217</td>
</tr>
<tr>
<td>T/cumul.</td>
<td>6/400</td>
<td>0/427</td>
<td>2/456</td>
<td>4/310</td>
<td>0/430</td>
<td>76/217</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T/constant</td>
<td>6/410</td>
<td>0/428</td>
<td>2/427</td>
<td>1/295</td>
<td>0/415</td>
<td>66/213</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S4/varying</td>
<td>0/458</td>
<td>-</td>
<td>30/289</td>
<td>1/421</td>
<td>109/199</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S4/cumul.</td>
<td>0/433</td>
<td>0/397</td>
<td>0/430</td>
<td>6/270</td>
<td>1/384</td>
<td>115/163</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S4/constant</td>
<td>0/448</td>
<td>0/401</td>
<td>2/397</td>
<td>4/255</td>
<td>1/368</td>
<td>100/162</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S5/varying</td>
<td>0/427</td>
<td>-</td>
<td>27/265</td>
<td>1/369</td>
<td>132/148</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S5/cumul.</td>
<td>0/418</td>
<td>0/379</td>
<td>0/244</td>
<td>1/315</td>
<td>126/118</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>S5/constant</td>
<td>0/436</td>
<td>2/359</td>
<td>0/231</td>
<td>1/315</td>
<td>116/118</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The column entries x/y in this table show (i) the number x of problems that were exclusively solved (i.e. proved or refuted) by an ATP system in a particular logic&domain and (ii) the average CPU time y in seconds needed by an ATP system for solving all problems in a particular logic&domain (the full 600s timeout was counted for each failing attempt).