Implementing and Evaluating Provers for First-order Modal Logics

Christoph Benzmüller[1], Jens Otten[2] and Thomas Raths[2]

[1]Freie Universität Berlin

[2]University of Potsdam

Motivation

First-order Modal Logics (FMLs)

$$p, q \quad ::= \quad P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \vee q) \mid \Box p \mid (\forall x p)$$

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

Talk Outline

Theory & implementation of new provers for FML:

- embedding into higher-order logic (LEO-II & Satallax) 2
- a connection calculus based prover (MleanCoP) 3
- a sequent calculus based prover (MleanSeP) 3
- a tableau based prover (MleanTAP) 4
- an instantiation based prover (f2p-MSPASS) 4

Moreover, we present

- a first comparative prover evaluation 1
- exploiting the new QMLTP library for FML 5

Experiment: $\underbrace{\textbf{580 problems} \times \textbf{5 logics} \times \textbf{3 domain conditions}}_{\textbf{8700 problems}} \times$ **6 provers** $\times$ **600s tmo**

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

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

Strongest Prover!

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

Logic/ Domain	f2p-MSPASS v3.0	MleanSeP v1.2	LEO-II v1.4.2	Satallax v2.2	MleanTAP v1.3	MleanCoP v1.2
K/varying	-	-	72	104	-	-
K/cumul.	70	121	89	122	-	-
K/constant	67	124	120	146	-	-
D/varying	-	-	128 81	113	100	179
D/cumul.	79	130	144 100	133	120	200
D/constant	76	134	167 135	160	135	217
T/varying	-	-	170 120	170	138	224
T/cumul.	105	163	190 139	192	160	249
T/constant	95	166	217 173	213	175	269
S4/varying	-	-	140	207	169	274
S4/cumul.	121	197	218 166	238	205	338
S4/constant	111	197	244 200	261	220	352
S5/varying	-	-	169	248	219	359
S5/cumul.	140	-	215	297	272	438
S5/constant	131	-	237	305	272	438

Strong improvement ($\geq$ 25%)

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

Logic/ Domain	ATP system f2p-MSPASS v3.0	MleanSeP v1.2	LEO-II v1.4.2	Satallax v2.2	MleanTAP v1.3	MleanCoP v1.2
K/varying	-	-	72	104	-	-
K/cumul.	70	121	89	122	-	-
K/constant	67	124	120	146	-	-
D/varying	-	-	128 81	113	100	179
D/cumul.	79	130	144 100	133	120	200
D/constant	76	134	167 135	160	135	217
T/varying	-	-	170 120	170	138	224
T/cumul.	105	163	190 139	192	160	249
T/constant	95	166	217 173	213	175	269
S4/varying	-	-	140	207	169	274
S4/cumul.	121	197	218 166	238	205	338
S4/constant	111	197	244 200	261	220	352
S5/varying	-	-	169	248	219	359
S5/cumul.	140	-	215	297	272	438
S5/constant	131	-	237	305	272	438

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

Embedding in HOL

Simple Types $\alpha ::= \iota \mid o \mid \alpha_1 \to \alpha_2$

Embedding in HOL

Simple Types $\alpha ::= \iota \mid o \mid \alpha_1 \to \alpha_2$

Individuals

Booleans (True and False)

Functions/Predicates

Embedding in HOL

Simple Types $\alpha ::= \iota \mid \mu \mid o \mid \alpha_1 \rightarrow \alpha_2$

Possible worlds → ι

Individuals → μ

Booleans (True and False) → o

Functions/Predicates → $\alpha_1 \rightarrow \alpha_2$

Embedding in HOL

HOL $\quad s,t \quad ::= \quad C_\alpha \mid x_\alpha \mid (\lambda x_\alpha . s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid (\neg_{o\to o}\, s_o)_o \mid (s_o \vee_{o\to o\to o} t_o)_o \mid (\forall x_\alpha t_o)_o$

Constant Symbols → C_α

Variable Symbols → x_α

Embedding in HOL

HOL

$$s, t \;::=\; C_\alpha \mid x_\alpha \mid (\lambda x_\alpha . s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid (\neg_{o\to o}\, s_o)_o \mid (s_o \vee_{o\to o\to o} t_o)_o \mid (\forall x_\alpha t_o)_o$$

Constant Symbols

Variable Symbols

Abstraction

Application

Embedding in HOL

HOL

$$s, t \quad ::= \quad C_\alpha \mid x_\alpha \mid (\lambda x_\alpha . s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o)_o \mid (s_o \vee_{o\to o\to o} t_o)_o \mid (\forall x_\alpha t_o)_o$$

Constant Symbols
Variable Symbols
Abstraction
Application
Logical Connectives

Embedding in HOL

HOL

$$s,t \quad ::= \quad C_\alpha \mid x_\alpha \mid (\lambda x_\alpha . s_\beta)_{\alpha\to\beta} \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid$$
$$(\neg_{o\to o}\, s_o)_o \mid (s_o \vee_{o\to o\to o} t_o)_o \mid (\forall x_\alpha t_o)_o$$

Embedding in HOL

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

Embedding in HOL

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

HOL (with Henkin semantics) is meanwhile very well understood

Embedding in HOL

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

HOL (with Henkin semantics) is meanwhile very well understood

HOL TPTP Infrastructure

Embedding in HOL

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

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

FML $\quad p, q ::= P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \vee q) \mid \Box p \mid (\forall x p)$

$$
\begin{array}{lll}
M, g, s \models \neg p & \text{iff} & \text{not } M, g, s \models p \\
M, g, s \models p \vee q & \text{iff} & M, g, s \models p \text{ or } M, g, s \models q \\
M, g, s \models \Box p & \text{iff} & M, g, u \models p \text{ for all } u \text{ with } R(s, u) \\
M, g, s \models \forall x p & \text{iff} & M, [d/x]g, s \models p \text{ for all } d \in D
\end{array}
$$

Embedding in HOL

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

FML $\quad p, q \quad ::= \quad P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \vee q) \mid \Box p \mid (\forall x p)$

$$
\begin{array}{lll}
M, g, s \models \neg p & \text{iff} & \text{not } M, g, s \models p \\
M, g, s \models p \vee q & \text{iff} & M, g, s \models p \text{ or } M, g, s \models q \\
M, g, s \models \Box p & \text{iff} & M, g, u \models p \text{ for all } u \text{ with } R(s, u) \\
M, g, s \models \forall x p & \text{iff} & M, [d/x]g, s \models p \text{ for all } d \in D
\end{array}
$$

FML in HOL:

$$
\begin{array}{lll}
\neg & = & \lambda p.\, \lambda w.\, \neg(pw) \\
\vee & = & \lambda p.\, \lambda q.\, \lambda w.\, (pw) \vee (qw) \\
\Box & = & \lambda p.\, \lambda w.\, \forall v (\neg(Rwv) \vee (pv)) \\
\forall & = & \lambda h.\, \lambda w.\, \forall x (hxw)
\end{array}
$$

Embedding in HOL

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

FML $\quad p, q \quad ::= \quad P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \vee q) \mid \Box p \mid (\forall x p)$

$$
\begin{array}{lll}
M, g, s \models \neg p & \text{iff} & \text{not } M, g, s \models p \\
M, g, s \models p \vee q & \text{iff} & M, g, s \models p \text{ or } M, g, s \models q \\
M, g, s \models \Box p & \text{iff} & M, g, u \models p \text{ for all } u \text{ with } R(s, u) \\
M, g, s \models \forall x p & \text{iff} & M, [d/x]g, s \models p \text{ for all } d \in D
\end{array}
$$

FML in HOL:

$$
\begin{array}{lll}
\neg & = & \lambda p. \lambda w. \neg(pw) \\
\vee & = & \lambda p. \lambda q. \lambda w. (pw) \vee (qw) \\
\Box & = & \lambda p. \lambda w. \forall v (\neg(Rwv) \vee (pv)) \\
\forall & = & \lambda h. \lambda w. \forall x (hxw)
\end{array}
$$

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

Embedding in HOL

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

FML $p, q \quad ::= \quad P(t_1, \ldots, t_n) \mid (\neg p) \mid (p \vee q) \mid \Box p \mid (\forall x p)$

$$
\begin{array}{lll}
M, g, s \models \neg p & \text{iff} & \text{not } M, g, s \models p \\
M, g, s \models p \vee q & \text{iff} & M, g, s \models p \text{ or } M, g, s \models q \\
M, g, s \models \Box p & \text{iff} & M, g, u \models p \text{ for all } u \text{ with } R(s, u) \\
M, g, s \models \forall x p & \text{iff} & M, [d/x]g, s \models p \text{ for all } d \in D
\end{array}
$$

FML in HOL:

$$
\begin{array}{lll}
\neg & = & \lambda p. \lambda w. \neg (p w) \\
\vee & = & \lambda p. \lambda q. \lambda w. (p w) \vee (q w) \\
\Box & = & \lambda p. \lambda w. \forall v (\neg (R w v) \vee (p v)) \\
\forall & = & \lambda h. \lambda w. \forall x (h x w)
\end{array}
$$

Meta-level notions: $\text{valid} \quad = \quad \lambda p. \forall w. p w$

Soundness & Completeness

Embedding in HOL

$$(\Diamond \exists x P f x \wedge \Box \forall y (\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$$

Embedding in HOL

$$(\Diamond \exists x P f x \wedge \Box \forall y (\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$$

$$\mathit{valid}\ (\Diamond \exists x P f x \wedge \Box \forall y (\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$$

Embedding in HOL

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

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

$$\Box \ = \lambda p.\lambda w.\forall v(\neg(Rwv) \vee (pv))$$

Embedding in HOL

$$(\Diamond\exists x P f x \wedge \Box\forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond\exists z Q z$$

$$\mathit{valid}\ (\Diamond\exists x P f x \wedge \Box\forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond\exists z Q z$$

$$\mathit{valid}\ (\Diamond\exists x P f x \wedge (\lambda w.\forall v(\neg(R w v) \vee (\forall y(\Diamond P y \Rightarrow Q y)\ v)))) \Rightarrow \Diamond\exists z Q z$$

$$\Box = \lambda p. \lambda w. \forall v(\neg(R w v) \vee (p v))$$

Embedding in HOL

$$(\Diamond\exists x P f x \wedge \Box\forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond\exists z Q z$$

$$valid\ (\Diamond\exists x P f x \wedge \Box\forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond\exists z Q z$$

$$valid\ (\Diamond\exists x P f x \wedge (\lambda w. \forall v(\neg(R w v) \vee (\forall y(\Diamond P y \Rightarrow Q y)\ v)))) \Rightarrow \Diamond\exists z Q z$$

...

$$\forall w(\neg\neg(\neg\neg\forall v(\neg R w v \vee \neg\neg\forall x \neg P(f x) v) \vee \neg\forall v(\neg R w v \vee \forall y(\neg\neg\forall u(\neg R v u \vee \neg P y u) \vee Q y v))) \vee \neg\forall v(\neg R w v \vee \neg\neg\forall z \neg Q z v))$$

Embedding in HOL

$$(\Diamond\exists x P f x \land \Box\forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond\exists z Q z$$

$$valid\ (\Diamond\exists x P f x \land \Box\forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond\exists z Q z$$

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

...

$$\forall w(\neg\neg(\neg\neg\forall v(\neg Rwv \lor \neg\neg\forall x\neg P(fx)v) \lor \neg\forall v(\neg Rwv \lor \forall y(\neg\neg\forall u(\neg Rvu \lor \neg Pyu) \lor Qyv))) \lor \neg\forall v(\neg Rwv \lor \neg\neg\forall z\neg Qzv))$$

Embedding in HOL

$$(\Diamond \exists x P f x \wedge \Box \forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$$

$$\textit{valid}\ (\Diamond \exists x P f x \wedge \Box \forall y(\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$$

$$\textit{valid}\ (\Diamond \exists x Pfx \wedge (\lambda w. \forall v(\neg(Rwv) \vee (\forall y(\Diamond Py \Rightarrow Qy)\ v)))) \Rightarrow \Diamond \exists z Qz$$

...

$$\forall w(\neg\neg(\neg\neg\forall v(\neg Rwv \vee \neg\neg\forall x \neg P(fx)v) \vee \neg\forall v(\neg Rwv \vee \forall y(\neg\neg\forall u(\neg Rvu \vee \neg Pyu) \vee Qyv))) \vee \neg\forall v(\neg Rwv \vee \neg\neg\forall z \neg Qzv))$$

Axiomatization of properties of accessibility relation R

Logic K: no axioms

Logic T: $(\textit{reflexive}\ R)$ — which expands into $\forall x\, Rxx$

Logic S4: $(\textit{reflexive}\ R) \wedge (\textit{symmetric}\ R) \wedge (\textit{transitive}\ R)$

Logic ... ...

Embedding in HOL

$$(\Diamond \exists x P f x \wedge \Box \forall y (\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$$

$$\mathit{valid}\ (\Diamond \exists x P f x \wedge \Box \forall y (\Diamond P y \Rightarrow Q y)) \Rightarrow \Diamond \exists z Q z$$

$$\mathit{valid}\ (\Diamond \exists x P f x \wedge (\lambda w. \forall v (\neg (R w v) \vee (\forall y (\Diamond P y \Rightarrow Q y)\ v)))) \Rightarrow \Diamond \exists z Q z$$

...

$$\forall w (\neg\neg(\neg\neg\forall v(\neg R w v \ \vee\ \neg\neg\forall x \neg P(f x) v) \ \vee\ \neg\forall v(\neg R w v \ \vee\ \forall y(\neg\neg\forall u(\neg R v u \vee \neg P y u) \vee Q y v))) \vee \neg\forall v(\neg R w v \vee \neg\neg\forall z \neg Q z v))$$

Axiomatization of properties of accessibility relation R

Logic K: no axioms

Logic T: $(\mathit{reflexive}\ R)$ — which expands into $\forall x\, R x x$

Logic S4: $(\mathit{reflexive}\ R) \wedge (\mathit{symmetric}\ R) \wedge (\mathit{transitive}\ R)$

Logic ... ...

This automates FML with constant domain semantics in HOL

Embedding in HOL

To obtain varying domain semantics:

- modify quantifier: $\forall = \lambda q \lambda w \forall x\, \text{ExistsInW} x w \Rightarrow q x w$
- 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: $\forall = \lambda q \lambda w \forall x \, \mathsf{ExistsInW} x w \Rightarrow q x w$
- add non-emptiness axiom: $\forall w \, \exists x \mathsf{ExistsInW} x w$
- add designation axioms for constants c: $\forall w \, \mathsf{ExistsInW} c w$
  (similar for function symbols)

To obtain cumulative domain semantics:

- add axiom: $\forall x \forall v \forall w \, \mathsf{ExistsInW} x v \wedge R v w \Rightarrow \mathsf{ExistsInW} x w$

Modal Sequent Calculus

- Extends the classical sequent calculus by modal rules for $\Box$ and $\Diamond$.

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

$$\frac{\Gamma, F \vdash \Delta}{\Gamma, \Box F \vdash \Delta}\ \Box\text{-}\mathit{left}$$

$$\frac{\Gamma \vdash F, \Delta}{\Gamma \vdash \Diamond F, \Delta}\ \Diamond\text{-}\mathit{right}$$

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

$$\frac{\Gamma, F \vdash \Delta}{\Gamma, \Box F \vdash \Delta}\ \Box\text{-}left \qquad \frac{\Gamma_{(\Box)} \vdash F, \Delta_{(\Diamond)}}{\Gamma \vdash \Box F, \Delta}\ \Box\text{-}right$$

$$\frac{\Gamma \vdash F, \Delta}{\Gamma \vdash \Diamond F, \Delta}\ \Diamond\text{-}right \qquad \frac{\Gamma_{(\Box)}, F \vdash \Delta_{(\Diamond)}}{\Gamma, \Diamond F \vdash \Delta}\ \Diamond\text{-}left$$

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

Modal Sequent Calculus

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

$$\frac{\Gamma, F \vdash \Delta}{\Gamma, \Box F \vdash \Delta}\ \Box\textit{-left} \qquad \frac{\Gamma_{(\Box)} \vdash F, \Delta_{(\Diamond)}}{\Gamma \vdash \Box F, \Delta}\ \Box\textit{-right}$$

$$\frac{\Gamma \vdash F, \Delta}{\Gamma \vdash \Diamond F, \Delta}\ \Diamond\textit{-right} \qquad \frac{\Gamma_{(\Box)}, F \vdash \Delta_{(\Diamond)}}{\Gamma, \Diamond F \vdash \Delta}\ \Diamond\textit{-left}$$

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).

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

$$\frac{\Gamma, F \vdash \Delta}{\Gamma, \Box F \vdash \Delta}\ \Box\textit{-left} \qquad \frac{\Gamma_{(\Box)} \vdash F, \Delta_{(\Diamond)}}{\Gamma \vdash \Box F, \Delta}\ \Box\textit{-right}$$

$$\frac{\Gamma \vdash F, \Delta}{\Gamma \vdash \Diamond F, \Delta}\ \Diamond\textit{-right} \qquad \frac{\Gamma_{(\Box)}, F \vdash \Delta_{(\Diamond)}}{\Gamma, \Diamond F \vdash \Delta}\ \Diamond\textit{-left}$$

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 Γ and Δ are deleted.

Modal Sequent Calculus – Example/Implementation

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

Modal Sequent Calculus – Example/Implementation

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

$$\dfrac{\dfrac{\dfrac{}{Pfd \vdash Pfd, Qfd}\ \textit{axiom}}{Pfd \vdash \Diamond Pfd, Qfd}\ \Diamond\textit{-right} \qquad \dfrac{}{Pfd, Qfd \vdash Qfd}\ \textit{axiom}}{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash Qfd}\ \Rightarrow\textit{-left}$$

$$\dfrac{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash Qfd}{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash \exists z\, Qz}\ \exists\textit{-right}\ (z \backslash fd)$$

$$\dfrac{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash \exists z\, Qz}{Pfd, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z\, Qz}\ \forall\textit{-left}\ (y \backslash fd)$$

$$\dfrac{Pfd, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z\, Qz}{\exists x\, Pfx, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z\, Qz}\ \exists\textit{-left}\ (x \backslash d)$$

$$\dfrac{\exists x\, Pfx, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z\, Qz}{\Diamond \exists x\, Pfx, \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z\, Qz}\ \Diamond\textit{-left}$$

$$\dfrac{\Diamond \exists x\, Pfx, \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z\, Qz}{\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z\, Qz}\ \wedge\textit{-left}$$

$$\dfrac{\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z\, Qz}{\vdash (\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz}\ \Rightarrow\textit{-right}$$

Modal Sequent Calculus – Example/Implementation

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

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Pfd \vdash Pfd, Qfd}\,\textit{axiom}}{Pfd \vdash \Diamond Pfd, Qfd}\,\Diamond\textit{-right} \qquad \dfrac{}{Pfd, Qfd \vdash Qfd}\,\textit{axiom}}{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash Qfd}\,\Rightarrow\textit{-left}}{Pfd, \Diamond Pfd \Rightarrow Qfd \vdash \exists z\, Qz}\,\exists\textit{-right}\ (z\backslash fd)}{Pfd, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z\, Qz}\,\forall\textit{-left}\ (y\backslash fd)}{\exists x\, Pfx, \forall y(\Diamond Py \Rightarrow Qy) \vdash \exists z\, Qz}\,\exists\textit{-left}\ (x\backslash d)}{\Diamond \exists x\, Pfx, \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z\, Qz}\,\Diamond\textit{-left}}{\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy) \vdash \Diamond \exists z\, Qz}\,\wedge\textit{-left}}{\vdash (\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz}\,\Rightarrow\textit{-right}$$

MleanSeP: implementation of the modal sequent calculus in PROLOG.

- analytic proof search with free variables and a dynamic Skolemization.
- available at http://www.leancop.de/mleansep/ (GPL license).

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 σ_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 $\{P(s_1, .., s_n), \neg P(t_1, .., , t_n)\}$.
- Connection corresponds to an axiom, if its literals unify under a first-order substitution σ_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
  $\mathcal{V}$: prefix variables (represent applications of $\Box$-*left* or $\Diamond$-*right*) and
  Π: 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 $\{P(s_1, .., s_n), \neg P(t_1, .., , t_n)\}$.
- Connection corresponds to an axiom, if its literals unify under a first-order substitution σ_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
  $\mathcal{V}$: prefix variables (represent applications of $\Box$-*left* or $\Diamond$-*right*) and
  Π: 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 σ_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
  $\mathcal{V}$: prefix variables (represent applications of $\Box$-*left* or $\Diamond$-*right*) and
  Π: 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 $\sigma_M : \mathcal{V} \to (\mathcal{V} \cup \Pi)^*$.

Connections and Prefixes – Example

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

Connections and Prefixes – Example

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

- sequent calculus

$$\dfrac{\dfrac{\overline{P \vdash}\ ?}{\Diamond P \vdash \Box P}\ \Diamond\textit{-left}}{\Diamond P \Rightarrow \Box P}\ \Rightarrow\textit{-right}$$

Connections and Prefixes – Example

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

- sequent calculus

$$\cfrac{\cfrac{\overline{P \vdash}\ ?}{\Diamond P \vdash \Box P}\ \Diamond\text{-}left}{\Diamond P \Rightarrow \Box P}\ \Rightarrow\text{-}right$$

- connection calculus

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

  $a \neq b$ $\rightsquigarrow$ prefixes not unifiable

  $\Longrightarrow$ formula not valid

Connections and Prefixes – Example

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

- sequent calculus

$$\dfrac{\dfrac{\overline{P \vdash}\;\; ?}{\Diamond P \vdash \Box P}\;\Diamond\text{-}left}{\Diamond P \Rightarrow \Box P}\;\Rightarrow\text{-}right$$

- connection calculus

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

  $a \neq b$ $\rightsquigarrow$ prefixes not unifiable

  $\Longrightarrow$ 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

$$\dfrac{\dfrac{\overline{P \vdash}\ ?}{\Diamond P \vdash \Box P}\ \Diamond\text{-}left}{\Diamond P \Rightarrow \Box P}\ \Rightarrow\text{-}right$$

- connection calculus

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

$a \neq b \leadsto$ prefixes not unifiable

$\Longrightarrow$ formula not valid

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

- sequent calculus

$$\dfrac{\dfrac{\overline{Q \vdash Q}\ axiom}{\Box Q \vdash \Diamond Q}\ \Box\text{-}left}{\Box Q \Rightarrow \Diamond Q}\ \Rightarrow\text{-}right$$

Connections and Prefixes – Example

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

- sequent calculus

$$\cfrac{\cfrac{\cfrac{}{P \vdash}\,?}{\Diamond P \vdash \Box P}\,\Diamond\text{-}left}{\Diamond P \Rightarrow \Box P}\,\Rightarrow\text{-}right$$

- connection calculus

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

  $a \neq b \;\rightsquigarrow$ prefixes not unifiable

  $\Longrightarrow$ formula not valid

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

- sequent calculus

$$\cfrac{\cfrac{\cfrac{}{Q \vdash Q}\,axiom}{\Box Q \vdash \Diamond Q}\,\Box\text{-}left}{\Box Q \Rightarrow \Diamond Q}\,\Rightarrow\text{-}right$$

- connection calculus

  connection $\{Q : V, \neg Q : W\}$

  $V = W \;\rightsquigarrow \sigma_M(V) = W$

  $\Longrightarrow$ formula valid

Connections and Prefixes – Example

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

- sequent calculus

$$\dfrac{\dfrac{\overline{P \vdash} \; ?}{\Diamond P \vdash \Box P}\ \Diamond\text{-}left}{\Diamond P \Rightarrow \Box P}\ \Rightarrow\text{-}right$$

- connection calculus

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

  $a \neq b \rightsquigarrow$ prefixes not unifiable

  $\Longrightarrow$ formula not valid

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

- sequent calculus

$$\dfrac{\dfrac{\overline{Q \vdash Q} \; axiom}{\Box Q \vdash \Diamond Q}\ \Box\text{-}left}{\Box Q \Rightarrow \Diamond Q}\ \Rightarrow\text{-}right$$

- connection calculus

  connection $\{Q : V, \neg Q : W\}$

  $V = W \rightsquigarrow \sigma_M(V) = W$

  $\Longrightarrow$ formula valid

Further restrictions on modal substitution σ_M (and σ_Q):

- induced reduction ordering has to be irreflexive,
- accessibility condition determines specific modal logic (D, T, S4, ...),
- domain constraint determines specific domain condition (constant, ...).

Prefixed Matrix

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

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.

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: $(\Diamond \exists x\, Pfx \wedge \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz$

- Prefixed matrix: $\{\{\neg Pfd : a_1\}, \{Py : V_1 V_2, \neg Qy : V_1\}, \{Qz : V_3\}\}$

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

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: $(\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz$

- Prefixed matrix: $\{\{\neg Pfd : a_1\}, \{Py : V_1 V_2, \neg Qy : V_1\}, \{Qz : V_3\}\}$

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

- Graphical representation:

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

Modal Connection Calculus – Example/Implementation

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

Modal Connection Calculus – Example/Implementation

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

$$\left[\; \left[\; \neg Pfd : a_1 \; \right] \; \left[\begin{array}{c} Py : V_1 V_2 \\ \neg Qy : V_1 \end{array} \right] \; \left[\; Qz : V_3 \; \right] \; \right]$$

Modal Connection Calculus – Example/Implementation

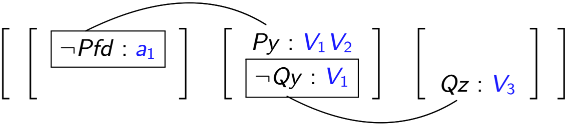
Example: $(\Diamond \exists x\, Pfx \wedge \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz$

$$\left[\ \left[\ \boxed{\neg Pfd : a_1}\ \right]\ \left[\ \begin{matrix} Py : V_1 V_2 \\ \neg Qy : V_1 \end{matrix}\ \right]\ \left[\ Qz : V_3\ \right]\ \right]$$

- with $\sigma_Q(y){=}fd$, $\sigma_M(V_1){=}a_1$, $\sigma_M(V_2){=}\varepsilon$

Modal Connection Calculus – Example/Implementation

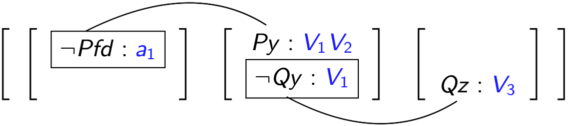
Example: $(\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz$



- 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: $(\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz$



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

Modal Connection Calculus – Example/Implementation

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

$$\left[\left[\boxed{\neg Pfd : a_1} \right] \left[\begin{array}{c} Py : V_1 V_2 \\ \boxed{\neg Qy : V_1} \end{array} \right] \left[Qz : V_3 \right] \right]$$

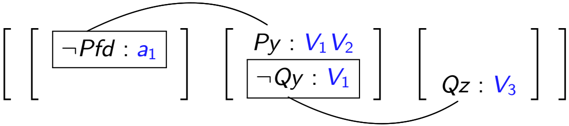
- with $\sigma_Q(y){=}fd$, $\sigma_Q(z){=}fd$, $\sigma_M(V_1){=}a_1$, $\sigma_M(V_2){=}\varepsilon$, and $\sigma_M(V_3){=}a_1$
  $\Longrightarrow$ formula is valid for T/S4 (constant/cumulative/varying domains).

MleanCoP: very compact implementation of modal connection calculus.

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

Modal Connection Calculus – Example/Implementation

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



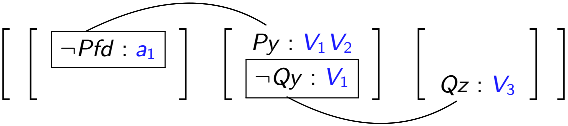
- with $\sigma_Q(y){=}fd$, $\sigma_Q(z){=}fd$, $\sigma_M(V_1){=}a_1$, $\sigma_M(V_2){=}\varepsilon$, and $\sigma_M(V_3){=}a_1$
  $\Longrightarrow$ formula is valid for T/S4 (constant/cumulative/varying domains).

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: $(\Diamond \exists x\, Pfx \wedge \Box \forall y (\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz$



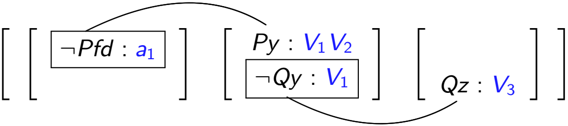
- with $\sigma_Q(y){=}fd$, $\sigma_Q(z){=}fd$, $\sigma_M(V_1){=}a_1$, $\sigma_M(V_2){=}\varepsilon$, and $\sigma_M(V_3){=}a_1$ $\Longrightarrow$ formula is valid for T/S4 (constant/cumulative/varying domains).

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: $(\Diamond \exists x\, Pfx \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z\, Qz$



- with $\sigma_Q(y){=}fd$, $\sigma_Q(z){=}fd$, $\sigma_M(V_1){=}a_1$, $\sigma_M(V_2){=}\varepsilon$, and $\sigma_M(V_3){=}a_1$
  $\Longrightarrow$ 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` (GNU GPL license).

MleanCoP – Source Code

- The source code of the leanCoP core prover for first-order classical logic.

```
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)
                                                                   .
```

MleanCoP – Source Code

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

```
prove([],_,_,_,[[],[]],_).
prove([Lit:Pre|Cla],Path,PathLim,Lem,[PreSet,FreeV],Set) :-
    \+ (member(LitC,[Lit:Pre|Cla]), member(LitP,Path), LitC==LitP),
    (-NegLit=Lit;-Lit=NegLit) ->
       ( member(LitL,Lem), Lit:Pre==LitL, PreSet3=[], FreeV3=[]
         ;
         member(NegL:PreN,Path), unify_with_occurs_check(NegL,NegLit),
         \+ \+ prefix_unify([Pre=PreN]), PreSet3=[Pre=PreN], FreeV3=[]
         ;
         lit(NegLit:PreN,FV:Cla1,Grnd1),
         \+ \+ prefix_unify([Pre=PreN]),
         ( Grnd1=g -> true ; length(Path,K), K<PathLim -> true ;
           \+ pathlim -> assert(pathlim), fail ),
         prove(Cla1,[Lit:Pre|Path],PathLim,Lem,[PreSet1,FreeV1],Set),
         PreSet3=[Pre=PreN|PreSet1], append(FreeV1,FV,FreeV3)
       ),
       ( member(cut,Set) -> ! ; true ),
       prove(Cla,Path,PathLim,[Lit:Pre|Lem],[PreSet2,FreeV2],Set),
       append(PreSet3,PreSet2,PreSet), append(FreeV2,FreeV3,FreeV).
```

Tableau Calculus

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

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 σ_Q / σ_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 σ_Q / σ_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 σ_Q/σ_M.
- Similar to connection calculus, but proof search not connection-driven.
- Particular logic specified by constraints on the modal substitution σ_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 σ_Q/σ_M.
- Similar to connection calculus, but proof search not connection-driven.
- Particular logic specified by constraints on the modal substitution σ_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 $\Box$ 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 σ_Q/σ_M.
- Similar to connection calculus, but proof search not connection-driven.
- Particular logic specified by constraints on the modal substitution σ_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 σ_Q/σ_M.
- Similar to connection calculus, but proof search not connection-driven.
- Particular logic specified by constraints on the modal substitution σ_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: $(\Diamond Pfd \wedge \Box\forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond\exists zQz$.

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: $(\Diamond Pfd \wedge \Box\forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond\exists z Qz$.

- first instance is not valid: $(\Diamond Pfd \wedge \Box(\Diamond Pa \Rightarrow Qa)) \Rightarrow \Diamond Qa$

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: $(\Diamond Pfd \wedge \Box \forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond \exists z Qz$.

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

$$(\Diamond Pfd \wedge \Box((\Diamond Pa \Rightarrow Qa) \wedge (\Diamond Pfd \Rightarrow Qfd))) \Rightarrow \Diamond(Qa \vee Qfd)$$

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: $(\Diamond Pfd \wedge \Box\forall y(\Diamond Py \Rightarrow Qy)) \Rightarrow \Diamond\exists z Qz$.

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

$$(\Diamond Pfd \wedge \Box((\Diamond Pa \Rightarrow Qa) \wedge (\Diamond Pfd \Rightarrow Qfd))) \Rightarrow \Diamond(Qa \vee Qfd)$$

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.
- 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 $\Box$ and $\Diamond$, 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)
% Refs     : [Sid09] T. Sider. Logic for Philosophy. Oxford, 2009.
%          : [Brc46] [1] R. C. Barcan. A functional calculus of first
%            order based on strict implication. Journal of Symbolic Logic
%            11:1-16, 1946.
% 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

Logic/ Domain	ATP system					
	f2p-MSPASS	MleanSeP	LEO-II	Satallax	MleanTAP	MleanCoP
K/varying	-	-	0/529	165/356	-	-
K/cumul.	88/363	4/471	0/511	50/349	-	-
K/constant	42/405	2/471	12/481	45/328	-	-
D/varying	-	-	0/519	0/477	0/492	293/173
D/cumul.	33/407	0/461	0/500	0/464	0/472	194/171
D/constant	33/411	0/462	2/466	0/425	0/456	167/169
T/varying	-	-	0/478	30/320	0/453	121/223
T/cumul.	6/400	0/427	2/456	4/310	0/430	76/217
T/constant	6/410	0/428	2/427	1/295	0/415	66/213
S4/varying	-	-	0/458	30/289	1/421	109/199
S4/cumul.	0/433	0/397	0/430	6/270	1/384	115/163
S4/constant	0/448	0/401	2/397	4/255	1/368	100/162
S5/varying	-	-	0/427	27/265	1/369	132/148
S5/cumul.	0/418	-	0/379	0/244	1/315	126/118
S5/constant	0/436	-	2/359	0/231	1/315	116/118

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).