Relation-Changing Modal Logics: Some Model and Proof Theoretic Aspects

Mauricio Martel

Universität Bremen, Germany

in collaboration with Carlos Areces, Raul Fervari, and Guillaume Hoffmann

Wormshop 2017
Moscow, Russia
• I don’t really work with provability logics or theories of arithmetic
• I don’t really work with provability logics or theories of arithmetic
• But... I’m interested in those topics pretty much since I first discovered modal logic and the incompleteness theorems
Disclaimer

- I don’t really work with provability logics or theories of arithmetic
- But... I’m interested in those topics pretty much since I first discovered modal logic and the incompleteness theorems
- I’m currently working with conservative extensions (as a decision problem) in description logics and guarded logics
I don’t really work with provability logics or theories of arithmetic.

But... I’m interested in those topics pretty much since I first discovered modal logic and the incompleteness theorems.

I’m currently working with conservative extensions (as a decision problem) in description logics and guarded logics.

But... I also work with modal logic! In particular, I’m interested in logics that can update the structure.
Disclaimer

• I don’t really work with provability logics or theories of arithmetic
• But... I’m interested in those topics pretty much since I first discovered modal logic and the incompleteness theorems
• I’m currently working with conservative extensions (as a decision problem) in description logics and guarded logics
• But... I also work with modal logic! In particular, I’m interested in logics that can update the structure
• This talk is about a particular family of logics that fall into that category
Modal logics are known to describe models
Modal Logics from a Semantic Perspective

- Modal logics are known to describe models
- Choose the right paintbrush:
  - $\Diamond \varphi$, $\Diamond \neg \varphi$
  - $E \varphi$
  - $\Diamond \geq_n \varphi$
  - $\Diamond^* \varphi$
  - ...

Now, what about operators that can modify models?
- Change the domain of the model
- Change the properties of the elements of the domain while we evaluate a formula
- Change the accessibility relation of a model while we evaluate a formula
Modal Logics from a Semantic Perspective

- Modal logics are known to describe models
- Choose the right paintbrush:
  - $\diamond \varphi$, $\diamond \neg \varphi$
  - $E \varphi$
  - $\diamond \geq n \varphi$
  - $\diamond^* \varphi$
  - ...
- Now, what about operators that can modify models?
Modal Logics from a Semantic Perspective

- Modal logics are known to describe models.
- Choose the right paintbrush:
  - $\Diamond \varphi$, $\Diamond \neg \varphi$
  - $E \varphi$
  - $\Diamond \geq n \varphi$
  - $\Diamond^* \varphi$
  - ...

- Now, what about operators that can modify models?
  - Change the domain of the model.
Modal Logics from a Semantic Perspective

- Modal logics are known to describe models
- Choose the right paintbrush:
  - $\Diamond \varphi, \Diamond \neg \varphi$
  - $E \varphi$
  - $\Diamond \geq n \varphi$
  - $\Diamond^* \varphi$
  - ...  
- Now, what about operators that can modify models?
  - Change the domain of the model
  - Change the properties of the elements of the domain while we evaluate a formula
Modal Logics from a Semantic Perspective

- Modal logics are known to describe models
- Choose the right paintbrush:
  - \( \Diamond \varphi, \Diamond \neg \varphi \)
  - \( E \varphi \)
  - \( \Diamond \geq n \varphi \)
  - \( \Diamond^* \varphi \)
  - ... 
- Now, what about operators that can modify models?
  - Change the domain of the model
  - Change the properties of the elements of the domain while we evaluate a formula
  - Change the accessibility relation of a model while we evaluate a formula
Logics that Change the Model

What about a *swapping* modal operator?

\[ \langle \text{sw} \rangle \lozenge \top \]

What happens when you add that to the basic modal logic?

\[ \langle \text{sw} \rangle \lozenge \top \]

\[ \circlearrowleft_{w} \rightarrow \circlearrowright_{v} \]

\[ \circlearrowleft_{w} \rightarrow \circlearrowright_{v} \]
Logics that Change the Model

What about a **swapping** modal operator?

\[
\langle \text{sw}\rangle \lozenge \top
\]

What about
- an edge-deleting modality?
- an edge-adding modality?
\( \mathcal{M}, w \models \langle gsb \rangle \varphi \iff \exists \text{ pair } (u, v) \text{ of } \mathcal{M} \text{ such that } \mathcal{M}^{\{u,v\}}, w \models \varphi, \)

where \( \mathcal{M}^{\{u,v\}} \) is \( \mathcal{M} \) without the edge \((u, v)\).

**Note:** \((u, v)\) can be anywhere in the model
Sabotage Modal Logic [van Benthem 05]

\[ \mathcal{M}, w \models \langle \text{gsb} \rangle \varphi \Leftrightarrow \exists \text{ pair } (u, v) \text{ of } \mathcal{M} \text{ such that } \mathcal{M}_{\{ (u, v) \}}, w \models \varphi, \]

where \( \mathcal{M}_{\{ (u, v) \}} \) is \( \mathcal{M} \) without the edge \((u, v)\).

**Note:** \((u, v)\) can be anywhere in the model

We are interested in operators that can modify the accessibility relation of a model.
Relation-Changing Modal Logics (RCML)

Remember the Basic Modal Logic ($\mathcal{ML}$)

- Syntax: propositional language + a modal operator $\Diamond$
- Semantics of $\Diamond \varphi$: traverse some edge, then evaluate $\varphi$
Relation-Changing Modal Logics (RCML)

Remember the Basic Modal Logic (ML)

- Syntax: propositional language + a modal operator ♦
- Semantics of ♦ϕ: traverse some edge, then evaluate ϕ

Now add new dynamic operators (sabotage, bridge, and swap):

- ⟨sb⟩ϕ: traverse some edge, delete it, then evaluate ϕ
- ⟨br⟩ϕ: add a new edge, traverse it, then evaluate ϕ
- ⟨sw⟩ϕ: traverse some edge, turn it around, then evaluate ϕ
- ⟨gsb⟩ϕ: delete some edge anywhere, then evaluate ϕ
- ⟨gbr⟩ϕ: add a new edge anywhere, then evaluate ϕ
- ⟨gsw⟩ϕ: swap an edge anywhere, then evaluate ϕ
Relation-Changing Modal Logics (RCML)

Remember the Basic Modal Logic ($\mathcal{ML}$)

- Syntax: propositional language + a modal operator $\Diamond$
- Semantics of $\Diamond \varphi$: traverse some edge, then evaluate $\varphi$

Now add new dynamic operators (sabotage, bridge, and swap):

- $\langle \text{sb} \rangle \varphi$: traverse some edge, delete it, then evaluate $\varphi$
Relation-Changing Modal Logics (RCML)

Remember the Basic Modal Logic ($\mathcal{ML}$)

- Syntax: propositional language + a modal operator $\Diamond$
- Semantics of $\Diamond \varphi$: traverse some edge, then evaluate $\varphi$

Now add new dynamic operators (sabotage, bridge, and swap):

- $\langle \text{sab}\rangle \varphi$: traverse some edge, delete it, then evaluate $\varphi$
- $\langle \text{br}\rangle \varphi$: add a new edge, traverse it, then evaluate $\varphi$
- $\langle \text{gsb}\rangle \varphi$: delete some edge anywhere, then evaluate $\varphi$
- $\langle \text{gbr}\rangle \varphi$: add a new edge anywhere, then evaluate $\varphi$
- $\langle \text{gsw}\rangle \varphi$: swap an edge anywhere, then evaluate $\varphi$
Relation-Changing Modal Logics (RCML)

Remember the Basic Modal Logic ($\mathcal{ML}$)

- Syntax: propositional language + a modal operator $\Diamond$
- Semantics of $\Diamond \varphi$: traverse some edge, then evaluate $\varphi$

Now add new dynamic operators (sabotage, bridge, and swap):

- $\langle \text{sb} \rangle \varphi$: traverse some edge, delete it, then evaluate $\varphi$
- $\langle \text{br} \rangle \varphi$: add a new edge, traverse it, then evaluate $\varphi$
- $\langle \text{sw} \rangle \varphi$: traverse some edge, turn it around, then evaluate $\varphi$
Relation-Changing Modal Logics (RCML)

Remember the Basic Modal Logic ($\mathcal{ML}$)
- Syntax: propositional language + a modal operator $\Diamond$
- Semantics of $\Diamond \varphi$: traverse some edge, then evaluate $\varphi$

Now add new dynamic operators (sabotage, bridge, and swap):
- $\langle sb \rangle \varphi$: traverse some edge, delete it, then evaluate $\varphi$
- $\langle br \rangle \varphi$: add a new edge, traverse it, then evaluate $\varphi$
- $\langle sw \rangle \varphi$: traverse some edge, turn it around, then evaluate $\varphi$
- $\langle gsb \rangle \varphi$: delete some edge anywhere, then evaluate $\varphi$
- $\langle gbr \rangle \varphi$: add a new edge anywhere, then evaluate $\varphi$
- $\langle gsw \rangle \varphi$: swap an edge anywhere, then evaluate $\varphi$
Relation-Changing Modal Logics (RCML)

Remember the Basic Modal Logic ($\mathcal{ML}$)
- Syntax: propositional language + a modal operator $\diamond$
- Semantics of $\diamond \varphi$: traverse some edge, then evaluate $\varphi$

Now add new dynamic operators (sabotage, bridge, and swap):
- $\langle sb \rangle \varphi$: traverse some edge, delete it, then evaluate $\varphi$
- $\langle br \rangle \varphi$: add a new edge, traverse it, then evaluate $\varphi$
- $\langle sw \rangle \varphi$: traverse some edge, turn it around, then evaluate $\varphi$
- $\langle gsb \rangle \varphi$: delete some edge anywhere, then evaluate $\varphi$
- $\langle gbr \rangle \varphi$: add a new edge anywhere, then evaluate $\varphi$
- $\langle gsw \rangle \varphi$: swap an edge anywhere, then evaluate $\varphi$

We call this family of logics Relation-Changing Modal Logics
Examples: Tree and Finite Model Properties

**Theorem**[ArecesFervariHoffmannWOLLIC12]
Relation-changing modal logics lack the tree model property.
Examples: Tree and Finite Model Properties

**Theorem** [ArecesFervariHoffmannWOLLIC12]
Relation-changing modal logics lack the tree model property.

**Proof of local operators**
1. $\Diamond \Diamond \top \land [sb] \Box \bot$ \hspace{1cm} $w$ is reflexive;
2. $\Box \bot \land \langle br \rangle \Box \bot$ \hspace{1cm} $w$ and $v \neq w$ are disconnected;
3. $p \land (\bigwedge_{1 \leq i \leq 3} \Box^i \neg p) \land \langle sw \rangle \Diamond \Diamond p$ \hspace{1cm} $w$ has a reflexive successor.
Examples: Tree and Finite Model Properties

Theorem [ArecesFervariHoffmannWOLLIC12]
Relation-changing modal logics lack the tree model property.

Proof of local operators
1. $\diamond\diamond T \land [sb]\Box \bot$ \hspace{1cm} $w$ is reflexive;
2. $\Box \bot \land \langle br \rangle \Box \bot$ \hspace{1cm} $w$ and $v \neq w$ are disconnected;
3. $p \land (\land_{1\leq i\leq 3} \Box i \neg p) \land \langle sw \rangle \diamond \diamond p$ \hspace{1cm} $w$ has a reflexive successor.

Theorem [MartelMScThesis15]
Relation-changing modal logics lack the finite model property.
Examples: Tree and Finite Model Properties

**Theorem** [ArecesFervariHoffmannWOLLIC12]
Relation-changing modal logics lack the tree model property.

**Proof of local operators**
1. $\Diamond\Diamond\top \land [sb]\Box\bot$ \hspace{1cm} $w$ is reflexive;
2. $\Box\bot \land \langle\text{br}\rangle\Box\bot$ \hspace{1cm} $w$ and $v \neq w$ are disconnected;
3. $p \land (\land_{1 \leq i \leq 3} \Box^i\neg p) \land \langle\text{sw}\rangle\Diamond\Diamond p$ \hspace{1cm} $w$ has a reflexive successor.

**Theorem** [MartelMScThesis15]
Relation-changing modal logics lack the finite model property.
Some Results about RCML

- Lack of tree-model property and finite model property (more expressivity than $\mathcal{ML}$)
- Incomparable among them in expressive power (using suitable notions of bisimulation)
- Model checking is \textit{PSPACE}-complete (via QBF reduction)
- The satisfiability problem is undecidable (via spy points and memory logic reduction)
- Standard translations into first-order logic

In this talk we show:
- translations into hybrid logics
- sound and complete (but non-terminating) tableaux methods
Some Results about RCML

• Lack of tree-model property and finite model property (more expressivity than $\mathcal{ML}$)
• Incomparable among them in expressive power (using suitable notions of bisimulation)
Some Results about RCML

- Lack of **tree-model property and finite model property** (more expressivity than $\mathcal{ML}$)
- Incomparable among them in **expressive power** (using suitable notions of bisimulation)
- **Model checking** is $\text{PSPACE}$-complete (via QBF reduction)
Some Results about RCML

- Lack of tree-model property and finite model property (more expressivity than $\mathcal{ML}$)
- Incomparable among them in expressive power (using suitable notions of bisimulation)
- Model checking is $\text{PSPACE}$-complete (via QBF reduction)
- The satisfiability problem is undecidable (via spy points and memory logic reduction)
Some Results about RCML

- Lack of tree-model property and finite model property (more expressivity than $\mathcal{ML}$)
- Incomparable among them in expressive power (using suitable notions of bisimulation)
- Model checking is $\text{PSpace}$-complete (via QBF reduction)
- The satisfiability problem is undecidable (via spy points and memory logic reduction)
- Standard translations into first-order logic
Some Results about RCML

- Lack of tree-model property and finite model property (more expressivity than $\mathcal{ML}$)
- Incomparable among them in expressive power (using suitable notions of bisimulation)
- Model checking is $\text{PSPACE}$-complete (via QBF reduction)
- The satisfiability problem is undecidable (via spy points and memory logic reduction)
- Standard translations into first-order logic

In this talk we show
- translations into hybrid logics
- sound and complete (but non-terminating) tableaux methods
Hybrid Logics

• The basic hybrid logic $\mathcal{HL}$ is obtained by adding a set $\text{NOM}$ of nominals to $\mathcal{ML}$. For $n \in \text{NOM}$, its valuation is a singleton set $V(n) = \{w\}$, for some state $w$. 
Hybrid Logics

- The basic hybrid logic $\mathcal{HL}$ is obtained by adding a set $\text{NOM}$ of nominals to $\mathcal{ML}$. For $n \in \text{NOM}$, its valuation is a singleton set $V(n) = \{w\}$, for some state $w$
- We have a satisfaction operator $n : \varphi$ with the usual semantics:

\[
\mathcal{M}, w \models n : \varphi \text{ iff } \mathcal{M}, v \models \varphi, \text{ where } V(n) = \{v\}
\]
Hybrid Logics

- The basic hybrid logic $\mathcal{HL}$ is obtained by adding a set NOM of nominals to $\mathcal{ML}$. For $n \in \text{NOM}$, its valuation is a singleton set $V(n) = \{w\}$, for some state $w$
- We have a satisfaction operator $n : \varphi$ with the usual semantics:
  $$\mathcal{M}, w \models n : \varphi \iff \mathcal{M}, v \models \varphi, \text{ where } V(n) = \{v\}$$
- And we also consider the down-arrow binder operator ↓:
  $$\langle W, R, V \rangle, w \models \downarrow n.\varphi \iff \langle W, R, V^w_n \rangle, w \models \varphi,$$
  where $V^w_n(n) = \{w\}$ and $V^w_n(m) = V(m)$, when $n \neq m$
Translations of RCML into Hybrid Logics

The translations are parametrized over a set of pair of nominals $S \subseteq \text{NOM} \times \text{NOM}$ that simulates the modification of edges.
Translations of RCML into Hybrid Logics

The translations are parametrized over a set of pair of nominals \( S \subseteq \text{NOM} \times \text{NOM} \) that simulates the modification of edges

Sabotage to Hybrid Logic

We define the translation \( (\_)_S \) from formulas of \( \mathcal{ML}(\langle \text{sb} \rangle) \) to formulas of \( \mathcal{HL}(:,\downarrow) \) as:

\[
(\Diamond \varphi)_S = \downarrow n. \Diamond \left( \bigwedge_{xy \in S} \neg (y \land n:x) \land (\varphi)_S \right)
\]

\[
(\langle \text{sb} \rangle \varphi)_S = \downarrow n. \Diamond \left( \bigwedge_{xy \in S} \neg (y \land n:x) \land \downarrow m. (\varphi)_{S \cup nm}' \right)
\]
Translations of RCML into Hybrid Logics

The translations are parametrized over a set of pair of nominals $S \subseteq \text{NOM} \times \text{NOM}$ that simulates the modification of edges

Sabotage to Hybrid Logic

We define the translation $(\_')_S$ from formulas of $\mathcal{ML}(\langle \text{sb} \rangle)$ to formulas of $\mathcal{HL}(\cdot, \downarrow)$ as:

$$(\Diamond \varphi)'_S = \downarrow n. \Diamond (\bigwedge_{xy \in S} \neg (y \land n:x) \land (\varphi)'_S)$$

$$(\langle \text{sb} \rangle \varphi)'_S = \downarrow n. \Diamond (\bigwedge_{xy \in S} \neg (y \land n:x) \land \downarrow m.(\varphi)'_{S \cup nm})$$

And for $\mathcal{ML}(\langle \text{gsb} \rangle)$ we translate into $\mathcal{HL}(E, \downarrow)$:

$$(\langle \text{gsb} \rangle \varphi)'_S = \downarrow k.E \downarrow n. \Diamond (\bigwedge_{xy \in S} \neg (y \land n:x) \land \downarrow m.k:(\varphi)'_{S \cup nm})$$
Translations of RCML into Hybrid Logics

The translations are parametrized over a set of pair of nominals $S \subseteq \text{NOM} \times \text{NOM}$ that simulates the modification of edges

Sabotage to Hybrid Logic

We define the translation $(\_ )'_{S}$ from formulas of $\mathcal{ML}(\langle \text{sb} \rangle )$ to formulas of $\mathcal{HL}(\cdot, \downarrow)$ as:

$$(\Diamond \varphi)'_{S} = \downarrow n. \Diamond (\bigwedge_{xy \in S} \neg(y \land n:x) \land (\varphi)'_{S})$$

$$(\langle \text{sb} \rangle \varphi)'_{S} = \downarrow n. \Diamond (\bigwedge_{xy \in S} \neg(y \land n:x) \land \downarrow m.(\varphi)'_{S \cup nm})$$

And for $\mathcal{ML}(\langle \text{gsb} \rangle )$ we translate into $\mathcal{HL}(E, \downarrow)$:

$$(\langle \text{gsb} \rangle \varphi)'_{S} = \downarrow k.E \downarrow n. \Diamond (\bigwedge_{xy \in S} \neg(y \land n:x) \land \downarrow m.k:(\varphi)'_{S \cup nm})$$

Translations for bridge and swap follow similar ideas (although for swap they are more involved)
Tableaux

- Check satisfiability of a formula by building a model
- Quite common procedure for modal logics in general
Tableaux

• Check satisfiability of a formula by building a model
• Quite common procedure for modal logics in general

Common features of tableaux for RCML

• Modal tableaux usually prefix formulas with constants to indicate where do formulas hold in a model
Tableaux

- Check satisfiability of a formula by building a model
- Quite common procedure for modal logics in general

Common features of tableaux for RCML

- Modal tableaux usually prefix formulas with constants to indicate where do formulas hold in a model
- Here we prefix formulas with one constant and a set of edge names. An edge name is a pair of constants
Tableaux

- Check satisfiability of a formula by building a model
- Quite common procedure for modal logics in general

Common features of tableaux for RCML

- Modal tableaux usually prefix formulas with constants to indicate where do formulas hold in a model
- Here we prefix formulas with one constant and a set of edge names. An edge name is a pair of constants
- Prefixed formulas are of the form: \((s, S) : \varphi\)
Tableaux

- Check satisfiability of a formula by building a model
- Quite common procedure for modal logics in general

Common features of tableaux for RCML

- Modal tableaux usually prefix formulas with constants to indicate where do formulas hold in a model
- Here we prefix formulas with one constant and a set of edge names. An edge name is a pair of constants
- Prefixed formulas are of the form: \((s, S) : \varphi\)

  “\(\varphi\) holds at the state referred to by \(s\) in the model variant described by the set of sabotaged/new/swapped edges \(S\)”
Common Tableaux Rules

- **Boolean rules**: decompose formulas, maintain prefix

\[
\frac{(n, X) : \varphi \land \psi}{(n, X) : \varphi} \quad (\land)
\]

\[
\frac{(n, X) : \varphi \land \psi}{(n, X) : \psi} \quad (\land)
\]

\[
\frac{(n, X) : \varphi \lor \psi}{(n, X) : \varphi \mid (n, X) : \psi} \quad (\lor)
\]
Common Tableaux Rules

- **Boolean rules**: decompose formulas, maintain prefix

  \[
  (n, X) : \varphi \land \psi \\
  \underline{\hspace{2cm}} (\land) \\
  (n, X) : \varphi \\
  (n, X) : \psi
  \]

  \[
  (n, X) : \varphi \lor \psi \\
  \underline{\hspace{2cm}} (\lor) \\
  (n, X) : \varphi \\
  (n, X) : \psi
  \]

- **Clashing rules**: atomic clash and “\( \neq \) versus equality” clash

  \[
  (n, X_1) : p \\
  (n, X_2) : \neg p \\
  \underline{\hspace{2cm}} (\bot_{atom})
  \]

  \[
  n \sim_{\Theta} m \\
  \underline{\hspace{2cm}} (\bot_{=})
  \]

  \[
  n \neq m \\
  \underline{\hspace{2cm}} (\bot_{\neq})
  \]
Common Tableaux Rules

- **Equational rules**: generate all formulas implied by constant equality

\[
\begin{align*}
\frac{\hat{R}nm}{\hat{R}\bar{n}\bar{m}} \quad (R\sim) \\
\frac{(n, X) : \varphi}{(\bar{n}, X) : \varphi} \quad (I\text{d})
\end{align*}
\]
Common Tableaux Rules

- **Equational rules**: generate all formulas implied by constant equality

\[
\frac{\dot{R}_nm}{\dot{R}\bar{n}\bar{m}} \quad (R\sim) \\
\frac{(n, X) : \varphi}{(\bar{n}, X) : \varphi} \quad (Id)
\]

- “**Unrestricted Blocking**” rule: saturate branch with equalities and inequalities between all pairs of constants

\[
\frac{n=m \mid n\neq m}{(ub)}
\]
Local Sabotage Tableaux

\[(n, S) : \diamond \varphi \quad \begin{array}{l}
\dot{R}_{nm} \\
nm \notin S \\
(m, S) : \varphi
\end{array} \quad (\diamond)\]

\[(n, S) : \Box \varphi \quad \begin{array}{l}
\dot{R}_{nm} \\
nm \notin S
\end{array} \quad (\Box)\]

\[(n, S) : \langle \text{sb} \rangle \varphi \quad \begin{array}{l}
\dot{R}_{nm} \\
nm \notin S \\
(m, S) : \varphi
\end{array} \quad (\langle \text{sb} \rangle)\]

\[(n, S) : [\text{sb}] \varphi \quad \begin{array}{l}
\dot{R}_{nm} \\
nm \notin S \\
(m, S \cup nm) : \varphi
\end{array} \quad ([\text{sb}])\]
Local Sabotage Tableaux

\[ (n, S) : \diamond \varphi \quad (\diamond) \]
\[ \dot{R}_{nm} \quad n \notin S \quad (m, S) : \varphi \]

\[ (n, S) : \langle \text{sb} \rangle \varphi \quad (\langle \text{sb} \rangle) \]
\[ \dot{R}_{nm} \quad n \notin S \quad (m, S) : \varphi \]

\[ (n, S) : [\text{sb}] \varphi \quad (\text{[sb]}) \]
\[ \dot{R}_{nm} \quad n \notin S \quad (m, S \cup nm) : \varphi \]

Tableaux rules for bridge and swap follow similar ideas (although for
swap they are more involved)
Proof Theory for RCML

Results obtained in [ArecesFervariHoffmannFroCoS13]

- **Sound** and **complete** calculus for relation-changing modal logics
Proof Theory for RCML

Results obtained in [ArecesFervariHoffmannFroCoS13]

- **Sound** and **complete** calculus for relation-changing modal logics
- Combining incomplete **tableaux** with **model checking** provides termination:
Proof Theory for RCML

Results obtained in [ArecesFervariHoffmannFroCoS13]

- **Sound** and **complete** calculus for relation-changing modal logics
- Combining incomplete **tableaux** with **model checking** provides termination:
  - run tableaux for $N$ steps
Proof Theory for RCML

Results obtained in [ArecesFervariHoffmannFroCoS13]

- **Sound** and **complete** calculus for relation-changing modal logics
- Combining incomplete **tableaux with model checking** provides termination:
  - run tableaux for \( N \) steps
  - if closed: UNSAT

Ideas used in the calculus are similar to ideas used in the translations of RCML into hybrid logics.
Proof Theory for RCML

Results obtained in [ArecesFervariHoffmannFroCoS13]

- **Sound** and **complete** calculus for relation-changing modal logics
- Combining incomplete **tableaux** with **model checking** provides termination:
  - run tableaux for $N$ steps
  - if closed: UNSAT
  - if open and input formula true in induced model: SAT
Proof Theory for RCML

Results obtained in [ArecesFervariHoffmannFroCoS13]
  - **Sound** and **complete** calculus for relation-changing modal logics
  - Combining incomplete **tableaux** with **model checking** provides termination:
    - run tableaux for \( N \) steps
    - if closed: UNSAT
    - if open and input formula true in induced model: SAT
    - otherwise: DON’T KNOW
Proof Theory for RCML

Results obtained in [ArecesFervariHoffmannFroCoS13]

- **Sound** and **complete** calculus for relation-changing modal logics
- Combining incomplete **tableaux** with **model checking** provides termination:
  - run tableaux for $N$ steps
  - if closed: UNSAT
  - if open and input formula true in induced model: SAT
  - otherwise: DON’T KNOW

Ideas used in the **calculus** are similar to ideas used in the **translations** of RCML into hybrid logics
Conclusions

RCML are a family of logics that can update the model.
Conclusions

RCML are a family of logics that can update the model.

**Dynamic epistemic logics** can update the model too!
- **Public Announcement Logic** (\(\mathcal{PAL}\)): deletes all states which do not satisfy certain public announcement.
- **Arrow Update Logic** (AUL): preserves the edges satisfying a pre and a post-condition.

We introduced logics that can update the accessibility relation.
- We presented translations of RCML into hybrid logics.
- We presented sound and complete tableaux methods.

Further work using hybrid logic techniques:
- Find axiomatizations.
- Compute interpolants.

Thanks!
Conclusions

RCML are a family of logics that can update the model

Dynamic epistemic logics can update the model too!
  - **Public Announcement Logic (PAL)**: deletes all states which do not satisfy certain public announcement
  - **Arrow Update Logic (AUL)**: preserves the edges satisfying a pre and a post-condition

We introduced logics that can **update the accessibility relation**
Conclusions

RCML are a family of logics that can update the model

**Dynamic epistemic logics** can update the model too!
- **Public Announcement Logic (\(PALT\)):** deletes all states which do not satisfy certain public announcement
- **Arrow Update Logic (AUL):** preserves the edges satisfying a pre and a post-condition

We introduced logics that can **update the accessibility relation**
- We presented **translations** of RCML into hybrid logics
- We presented sound and complete **tableaux methods**
Conclusions

RCML are a family of logics that can update the model

Dynamic epistemic logics can update the model too!

- Public Announcement Logic ($\mathcal{PAL}$): deletes all states which do not satisfy certain public announcement
- Arrow Update Logic (AUL): preserves the edges satisfying a pre and a post-condition

We introduced logics that can update the accessibility relation

- We presented translations of RCML into hybrid logics
- We presented sound and complete tableaux methods

Further work using hybrid logic techniques

- Find axiomatizations
- Compute interpolants
Conclusions

RCML are a family of logics that can update the model.

Dynamic epistemic logics can update the model too!
- Public Announcement Logic (PAL): deletes all states which do not satisfy certain public announcement.
- Arrow Update Logic (AUL): preserves the edges satisfying a pre and a post-condition.

We introduced logics that can update the accessibility relation.
- We presented translations of RCML into hybrid logics.
- We presented sound and complete tableaux methods.

Further work using hybrid logic techniques.
- Find axiomatizations.
- Compute interpolants.

Thanks!