Undecidability of Relation-Changing Modal Logics

Carlos Areces[1,2], Raul Fervari[1,2], Guillaume Hoffmann[1,2] & Mauricio Martel[3]

[1] FaMAF, Universidad Nacional de Córdoba, Argentina
[2] CONICET, Argentina
[3] Universität Bremen, Germany

DaLí 2017, Brasília, Brazil

Modal logics: *"we like to talk about models"*

- Modal logics are known to describe models.
- Choose the right paintbrush:
  - $\Diamond\varphi$, $\Diamond^{-}\varphi$
  - $\mathsf{E}\varphi$
  - $\Diamond_{\geq n}\varphi$
  - $\Diamond^{*}\varphi$
  - ...
- 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 are evaluating a formula.
  - Evaluate φ after deleting/adding/swapping around an edge.

Logics that change the model 1/2

What about a swapping modal operator?

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

Logics that change the model 2/2

What about:

- an edge-deleting modality?

Logics that change the model 2/2

What about:

- an edge-deleting modality?
- an edge-adding modality?

Sabotage Modal Logic [van Benthem 2002]

$$\mathrm{M}, w \models \langle \mathrm{gsb} \rangle \varphi \quad \text{iff} \quad \exists \text{ pair } (u, v) \text{ of M such that } \mathrm{M}^{-}_{\{(u,v)\}}, w \models \varphi,$$

where $\mathrm{M}^{-}_{\{(u,v)\}}$ is M without the edge (u, v).

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

Sabotage Modal Logic [van Benthem 2002]

$$\mathsf{M}, w \models \langle \mathrm{gsb} \rangle \varphi \quad \text{iff} \quad \exists \text{ pair } (u, v) \text{ of } \mathsf{M} \text{ such that } \mathsf{M}^{-}_{\{(u,v)\}}, w \models \varphi,$$

where $\mathsf{M}^{-}_{\{(u,v)\}}$ is M without the edge (u, v).

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

What we know [Löding & Rohde 03]:

- Model checking is PSPACE-complete.
- Satisfiability is undecidable (multi-modal case, reduction from PCP).

Epistemic Operators

- Those are operators that also modify models!

Epistemic Operators

- Those are operators that also modify models!
- $[!\psi]\varphi$: announce that if ψ is true, eliminate states of the model where $\neg\psi$ holds (Public Announcement Logic) [Plaza 89].

Epistemic Operators

- Those are operators that also modify models!
- $[!\psi]\varphi$: announce that if ψ is true, eliminate states of the model where $\neg\psi$ holds (Public Announcement Logic) [Plaza 89].
- $\Diamond\varphi$: there is a $\Diamond$-free announcement ψ such that $[!\psi]\varphi$ holds (Arbitrary Public Announcement Logic) [Balbiani et al. 07].

Epistemic Operators

- Those are operators that also modify models!
- $[!\psi]\varphi$: announce that if ψ is true, eliminate states of the model where $\neg\psi$ holds (Public Announcement Logic) [Plaza 89].
- $\Diamond\varphi$: there is a $\Diamond$-free announcement ψ such that $[!\psi]\varphi$ holds (Arbitrary Public Announcement Logic) [Balbiani et al. 07].
- In some way these operators are deleting states.

Epistemic Operators

- Those are operators that also modify models!
- $[!\psi]\varphi$: announce that if ψ is true, eliminate states of the model where $\neg\psi$ holds (Public Announcement Logic) [Plaza 89].
- $\Diamond\varphi$: there is a $\Diamond$-free announcement ψ such that $[!\psi]\varphi$ holds (Arbitrary Public Announcement Logic) [Balbiani et al. 07].
- In some way these operators are deleting states.
- We will focus on operators that modify the accesibility relation.

Meet the operators

Remember the Basic Modal Logic (ML).

- Syntax: propositional language + a modal operator $\Diamond$.

Meet the operators

Remember the Basic Modal Logic (ML).

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

Meet the operators

Remember the Basic Modal Logic (ML).

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

Meet the operators

Remember the Basic Modal Logic (ML).

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

Now add new dynamic operators:

- Semantics of global/local **sw**ap, **sab**otage and **br**idge:

Meet the operators

Remember the Basic Modal Logic (ML).

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

Now add new dynamic operators:

- Semantics of global/local **sw**ap, **sab**otage and **br**idge:
  - $\langle \mathsf{sw} \rangle \varphi$: traverse some edge, turn it around, then evaluate φ.

Meet the operators

Remember the Basic Modal Logic (ML).

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

Now add new dynamic operators:

- Semantics of global/local **sw**ap, **sab**otage and **br**idge:
  - $\langle \mathrm{sw} \rangle\varphi$: traverse some edge, turn it around, then evaluate φ.
  - $\langle \mathrm{gsw} \rangle\varphi$: turn around some edge anywhere, then evaluate φ.

Meet the operators

Remember the Basic Modal Logic (ML).

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

Now add new dynamic operators:

- Semantics of global/local **sw**ap, **sab**otage and **br**idge:
  - $\langle \mathsf{sw} \rangle \varphi$: traverse some edge, turn it around, then evaluate φ.
  - $\langle \mathsf{gsw} \rangle \varphi$: turn around some edge anywhere, then evaluate φ.
  - $\langle \mathsf{sb} \rangle \varphi$: traverse some edge, delete it, then evaluate φ.

Meet the operators

Remember the Basic Modal Logic (ML).

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

Now add new dynamic operators:

- Semantics of global/local **sw**ap, **sab**otage and **br**idge:
  - $\langle \mathsf{sw} \rangle\varphi$: traverse some edge, turn it around, then evaluate φ.
  - $\langle \mathsf{gsw} \rangle\varphi$: turn around some edge anywhere, then evaluate φ.
  - $\langle \mathsf{sb} \rangle\varphi$: traverse some edge, delete it, then evaluate φ.
  - $\langle \mathsf{gsb} \rangle\varphi$: delete some edge anywhere, then evaluate φ.

Meet the operators

Remember the Basic Modal Logic (ML).

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

Now add new dynamic operators:

- Semantics of global/local **sw**ap, **sab**otage and **br**idge:
  - $\langle \mathsf{sw} \rangle\varphi$: traverse some edge, turn it around, then evaluate φ.
  - $\langle \mathsf{gsw} \rangle\varphi$: turn around some edge anywhere, then evaluate φ.
  - $\langle \mathsf{sb} \rangle\varphi$: traverse some edge, delete it, then evaluate φ.
  - $\langle \mathsf{gsb} \rangle\varphi$: delete some edge anywhere, then evaluate φ.
  - $\langle \mathsf{br} \rangle\varphi$: add a new edge, traverse it, then evaluate φ.

Meet the operators

Remember the Basic Modal Logic (ML).

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

Now add new dynamic operators:

- Semantics of global/local **sw**ap, **sab**otage and **br**idge:
  - $\langle \mathrm{sw} \rangle\varphi$: traverse some edge, turn it around, then evaluate φ.
  - $\langle \mathrm{gsw} \rangle\varphi$: turn around some edge anywhere, then evaluate φ.
  - $\langle \mathrm{sb} \rangle\varphi$: traverse some edge, delete it, then evaluate φ.
  - $\langle \mathrm{gsb} \rangle\varphi$: delete some edge anywhere, then evaluate φ.
  - $\langle \mathrm{br} \rangle\varphi$: add a new edge, traverse it, then evaluate φ.
  - $\langle \mathrm{gbr} \rangle\varphi$: add a new edge, then evaluate φ.

Very expressive languages

- These languages have been extensively investigated [Ferv14].

Very expressive languages

- These languages have been extensively investigated [Ferv14].
- More expressive than ML [AFH12,14].

Very expressive languages

- These languages have been extensively investigated [Ferv14].
- More expressive than ML [AFH12,14].
- Model checking is PSPACE-complete. [AFH12].

Very expressive languages

- These languages have been extensively investigated [Ferv14].
- More expressive than ML [AFH12,14].
- Model checking is PSPACE-complete. [AFH12].
- They are fragments of FOL [AFH14,15].

Very expressive languages

- These languages have been extensively investigated [Ferv14].
- More expressive than ML [AFH12,14].
- Model checking is PSPACE-complete. [AFH12].
- They are fragments of FOL [AFH14,15].
- Non-terminating tableaux systems [AFH13].

Very expressive languages

- These languages have been extensively investigated [Ferv14].
- More expressive than ML [AFH12,14].
- Model checking is PSPACE-complete. [AFH12].
- They are fragments of FOL [AFH14,15].
- Non-terminating tableaux systems [AFH13].
- Some undecidability results were known [AFH14].

Very expressive languages

- These languages have been extensively investigated [Ferv14].
- More expressive than ML [AFH12,14].
- Model checking is PSPACE-complete. [AFH12].
- They are fragments of FOL [AFH14,15].
- Non-terminating tableaux systems [AFH13].
- Some undecidability results were known [AFH14].
- Translations into Hybrid Logic [AFHM14].

Contributions

- We found connections between relation-changing operators and other dynamics operators (e.g. hybrid and memory logics).

Contributions

- We found connections between relation-changing operators and other dynamics operators (e.g. hybrid and memory logics).
- We can simulate some notions of binding by adding/deleting/swapping edges.

Contributions

- We found connections between relation-changing operators and other dynamics operators (e.g. hybrid and memory logics).
- We can simulate some notions of binding by adding/deleting/swapping edges.
- Then, we prove undecidability of RCML by using a spy point-like technique + reduction from the undecidable satisfiability problem of $\mathsf{ML}(ⓡ, ⓚ)$.

Contributions

- We found connections between relation-changing operators and other dynamics operators (e.g. hybrid and memory logics).
- We can simulate some notions of binding by adding/deleting/swapping edges.
- Then, we prove undecidability of RCML by using a spy point-like technique + reduction from the undecidable satisfiability problem of ML(ⓡ, ⓚ).
- Two steps:

Contributions

- We found connections between relation-changing operators and other dynamics operators (e.g. hybrid and memory logics).
- We can simulate some notions of binding by adding/deleting/swapping edges.
- Then, we prove undecidability of RCML by using a spy point-like technique + reduction from the undecidable satisfiability problem of ML(ⓡ, ⓚ).
- Two steps:
  - Adapt the undecidability proof of $\mathcal{ALC}$self to mono-modal memory logic.

Contributions

- We found connections between relation-changing operators and other dynamics operators (e.g. hybrid and memory logics).
- We can simulate some notions of binding by adding/deleting/swapping edges.
- Then, we prove undecidability of RCML by using a spy point-like technique + reduction from the undecidable satisfiability problem of ML(ⓡ, ⓚ).
- Two steps:
  - Adapt the undecidability proof of $\mathcal{ALC}$self to mono-modal memory logic.
  - Give a satisfiability preserving translation from memory logic into relation-changing logics.

Memory Logics in a nutshell

Models in ML(ⓡ, ⓚ) are extensions of classic Kripke models with a memory:

- $\mathrm{M} = \langle W, R, V, S \rangle$, with $S \subseteq W$.

Memory Logics in a nutshell

Models in ML($ⓡ$, $ⓚ$) are extensions of classic Kripke models with a memory:

- $\mathrm{M} = \langle W, R, V, S \rangle$, with $S \subseteq W$.

In addition to ML, the memory logic ML($ⓡ$, $ⓚ$) has two new operators:

- Remember: $\langle W, R, V, S \rangle, w \models ⓡ\varphi$ iff $\langle W, R, V, S \cup \{w\} \rangle, w \models \varphi$.

Memory Logics in a nutshell

Models in $\mathcal{ML}(ⓡ, ⓚ)$ are extensions of classic Kripke models with a memory:

- $\mathsf{M} = \langle W, R, V, S \rangle$, with $S \subseteq W$.

In addition to ML, the memory logic $\mathcal{ML}(ⓡ, ⓚ)$ has two new operators:

- Remember: $\langle W, R, V, S \rangle, w \models ⓡ\varphi$ iff $\langle W, R, V, S \cup \{w\} \rangle, w \models \varphi$.
- Known: $\langle W, R, V, S \rangle, w \models ⓚ$ iff $w \in S$.

Memory Logics in a nutshell

Models in $\mathcal{ML}(\text{ⓡ}, \text{ⓚ})$ are extensions of classic Kripke models with a memory:

- $\mathcal{M} = \langle W, R, V, S \rangle$, with $S \subseteq W$.

In addition to $\mathcal{ML}$, the memory logic $\mathcal{ML}(\text{ⓡ}, \text{ⓚ})$ has two new operators:

- Remember: $\langle W, R, V, S \rangle, w \models \text{ⓡ}\varphi$ iff $\langle W, R, V, S \cup \{w\} \rangle, w \models \varphi$.
- Known: $\langle W, R, V, S \rangle, w \models \text{ⓚ}$ iff $w \in S$.

Observation

ⓡ can be seen as marking a state as visited, and ⓚ as checking if a state has been visited.

Memory Logics in a nutshell

Models in $\mathcal{ML}(ⓡ, ⓚ)$ are extensions of classic Kripke models with a memory:

- $\mathcal{M} = \langle W, R, V, S \rangle$, with $S \subseteq W$.

In addition to $\mathcal{ML}$, the memory logic $\mathcal{ML}(ⓡ, ⓚ)$ has two new operators:

- Remember: $\langle W, R, V, S \rangle, w \models ⓡ\varphi$ iff $\langle W, R, V, S \cup \{w\} \rangle, w \models \varphi$.
- Known: $\langle W, R, V, S \rangle, w \models ⓚ$ iff $w \in S$.

Observation

ⓡ can be seen as marking a state as visited, and ⓚ as checking if a state has been visited.

Proposition

The satisfiability problem of the memory logic $\mathcal{ML}(ⓡ, ⓚ)$ is undecidable.

Undecidability proofs

Each translation from ML(ⓡ, ⓚ) into relation-changing logics proceeds in two steps:

- A fixed part called *Struct*, enforces constraints on the structure of the model.

Undecidability proofs

Each translation from ML(ⓡ, ⓚ) into relation-changing logics proceeds in two steps:

- A fixed part called *Struct*, enforces constraints on the structure of the model.
- The second part is defined inductively on ML(ⓡ, ⓚ)-formulas, and uses the structure provided by *Struct* to simulate the ⓡ and ⓚ operators.

Undecidability proofs

Each translation from ML(ⓡ, ⓚ) into relation-changing logics proceeds in two steps:

- A fixed part called *Struct*, enforces constraints on the structure of the model.
- The second part is defined inductively on ML(ⓡ, ⓚ)-formulas, and uses the structure provided by *Struct* to simulate the ⓡ and ⓚ operators.

Theorem

The satisfiability problem of the six RCML is undecidable.

Undecidability proofs

Each translation from $\mathrm{ML}(ⓡ, ⓚ)$ into relation-changing logics proceeds in two steps:

- A fixed part called *Struct*, enforces constraints on the structure of the model.
- The second part is defined inductively on $\mathrm{ML}(ⓡ, ⓚ)$-formulas, and uses the structure provided by *Struct* to simulate the ⓡ and ⓚ operators.

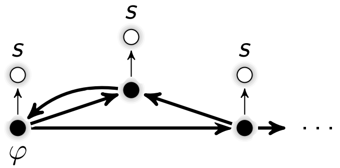
Theorem

The satisfiability problem of the six RCML is undecidable.

Proof

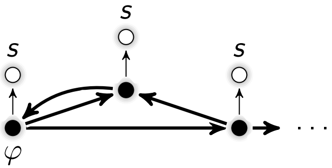
Satisfiability problem of $\mathrm{ML}(ⓡ, ⓚ) \Rightarrow$ satisfiability problem of $\mathrm{ML}(\blacklozenge)$, with $\blacklozenge \in \{\langle \mathrm{sb} \rangle, \langle \mathrm{gsb} \rangle, \langle \mathrm{br} \rangle, \langle \mathrm{gbr} \rangle, \langle \mathrm{sw} \rangle, \langle \mathrm{gsw} \rangle\}$.

Encoding ML(ⓡ, ⓚ) with global sabotage



$$Struct_{\langle \mathrm{gsb} \rangle}(\varphi) = \quad \neg s \wedge \bigwedge_{0 \leq i \leq \mathrm{md}(\varphi)} \Box^i(\neg s \ \rightarrow \Diamond s)$$

Encoding ML(ⓡ, ⓚ) with global sabotage

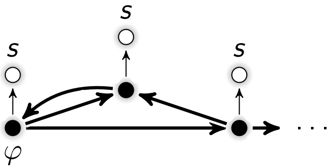


$$Struct_{\langle \mathrm{gsb} \rangle}(\varphi) = \neg s \wedge \bigwedge_{0 \leq i \leq \mathrm{md}(\varphi)} \Box^{i}(\neg s \rightarrow \Diamond s)$$

Let φ be a ML(ⓡ, ⓚ)-formula, we define the translation into ML($\langle$gsb$\rangle$):

$$\begin{array}{lcl} \mathrm{Tr}_{\langle \mathrm{gsb} \rangle}(ⓚ) & = & \neg \Diamond s \\ \mathrm{Tr}_{\langle \mathrm{gsb} \rangle}(\Diamond \varphi) & = & \Diamond(\neg s \wedge \mathrm{Tr}_{\langle \mathrm{gsb} \rangle}(\varphi)) \\ \mathrm{Tr}_{\langle \mathrm{gsb} \rangle}(ⓡ \varphi) & = & \langle \mathrm{gsb} \rangle(\neg \Diamond s \wedge \mathrm{Tr}_{\langle \mathrm{gsb} \rangle}(\varphi)) \end{array}$$

Encoding ML(ⓡ, ⓚ) with global sabotage



$$Struct_{\langle gsb \rangle}(\varphi) = \neg s \wedge \bigwedge_{0 \leq i \leq \mathrm{md}(\varphi)} \Box^i(\neg s \rightarrow \Diamond s)$$

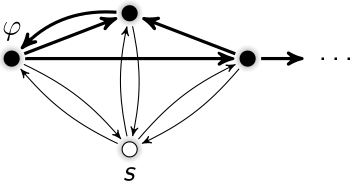
Let φ be a ML(ⓡ, ⓚ)-formula, we define the translation into ML($\langle$gsb$\rangle$):

$$\begin{array}{lcl} \mathrm{Tr}_{\langle gsb \rangle}(ⓚ) & = & \neg\Diamond s \\ \mathrm{Tr}_{\langle gsb \rangle}(\Diamond\varphi) & = & \Diamond(\neg s \wedge \mathrm{Tr}_{\langle gsb \rangle}(\varphi)) \\ \mathrm{Tr}_{\langle gsb \rangle}(ⓡ\varphi) & = & \langle gsb \rangle(\neg\Diamond s \wedge \mathrm{Tr}_{\langle gsb \rangle}(\varphi)) \end{array}$$

Then,

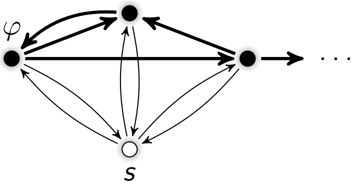
φ is satisfiable $\Leftrightarrow$ ($Struct_{\langle gsb \rangle}(\varphi) \wedge \mathrm{Tr}_{\langle gsb \rangle}(\varphi)$) is satisfiable

Encoding ML(ⓡ, ⓚ) with local sabotage



$$
\begin{aligned}
Struct_{\langle \mathrm{sb} \rangle} = \quad & s \wedge \Box\neg s \ \wedge \ \Box\Diamond s \ \wedge \ [\mathrm{sb}][\mathrm{sb}](s \rightarrow \Box\Diamond s) \\
& \wedge \Box[\mathrm{sb}](s \rightarrow \Diamond\Box\neg s) \\
& \wedge \Box\Box(\neg s \rightarrow \Diamond s) \\
& \wedge \Box[\mathrm{sb}](s \rightarrow [\mathrm{sb}](\Box\neg s \rightarrow \Box\Box(s \rightarrow \Box\Diamond s))) \\
& \wedge \Box[\mathrm{sb}](s \rightarrow \Box(\Box\neg s \rightarrow \Box\Box(s \rightarrow \Diamond\Box\neg s))) \\
& \wedge \Box\Box\Box(s \rightarrow \Box\Diamond s) \ \wedge \ \Box\Box[\mathrm{sb}](s \rightarrow \Diamond\Box\neg s)
\end{aligned}
$$

Encoding ML(ⓡ, ⓚ) with local sabotage

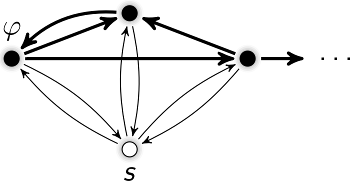


$$\begin{aligned}
Struct_{\langle sb\rangle} = \quad & s \wedge \Box\neg s \;\wedge\; \Box\Diamond s \;\wedge\; [sb][sb](s \to \Box\Diamond s) \\
& \wedge\, \Box[sb](s \to \Diamond\Box\neg s) \\
& \wedge\, \Box\Box(\neg s \to \Diamond s) \\
& \wedge\, \Box[sb](s\to[sb](\Box\neg s\to\Box\Box(s\to\Box\Diamond s))) \\
& \wedge\, \Box[sb](s\to\Box(\Box\neg s\to\Box\Box(s\to\Diamond\Box\neg s))) \\
& \wedge\, \Box\Box\Box(s \to \Box\Diamond s) \;\wedge\; \Box\Box[sb](s \to \Diamond\Box\neg s)
\end{aligned}$$

$\mathrm{Tr}_{\langle sb\rangle}(\varphi) = \Diamond(\varphi)'$, with:

$$\begin{aligned}
(\text{ⓚ})' &= \neg\Diamond s \\
(\Diamond\psi)' &= \Diamond(\neg s \wedge (\psi)') \\
(\text{ⓡ}\,\psi)' &= \langle sb\rangle(s \wedge \langle sb\rangle(\neg\Diamond s \wedge (\psi)'))
\end{aligned}$$

Encoding ML(ⓡ, ⓚ) with local sabotage



$$
\begin{aligned}
Struct_{\langle \mathrm{sb}\rangle} = \quad & s \wedge \Box\neg s \;\wedge\; \Box\Diamond s \;\wedge\; [\mathrm{sb}][\mathrm{sb}](s \rightarrow \Box\Diamond s) \\
& \wedge\, \Box[\mathrm{sb}](s \rightarrow \Diamond\Box\neg s) \\
& \wedge\, \Box\Box(\neg s \rightarrow \Diamond s) \\
& \wedge\, \Box[\mathrm{sb}](s \rightarrow [\mathrm{sb}](\Box\neg s \rightarrow \Box\Box(s \rightarrow \Box\Diamond s))) \\
& \wedge\, \Box[\mathrm{sb}](s \rightarrow \Box(\Box\neg s \rightarrow \Box\Box(s \rightarrow \Diamond\Box\neg s))) \\
& \wedge\, \Box\Box\Box(s \rightarrow \Box\Diamond s) \;\wedge\; \Box\Box[\mathrm{sb}](s \rightarrow \Diamond\Box\neg s)
\end{aligned}
$$

$\mathrm{Tr}_{\langle \mathrm{sb}\rangle}(\varphi) = \Diamond(\varphi)'$, with:

$$
\begin{aligned}
(ⓚ)' \quad &= \quad \neg\Diamond s \\
(\Diamond\psi)' \quad &= \quad \Diamond(\neg s \wedge (\psi)') \\
(ⓡ\psi)' \quad &= \quad \langle \mathrm{sb}\rangle(s \wedge \langle \mathrm{sb}\rangle(\neg\Diamond s \wedge (\psi)'))
\end{aligned}
$$

Then,

φ is satisfiable iff $(Struct_{\langle \mathrm{sb}\rangle} \wedge \mathrm{Tr}_{\langle \mathrm{sb}\rangle}(\varphi))$ is satisfiable

Other encodings

- Similar translations for the rest of logics.

Other encodings

- Similar translations for the rest of logics.
- For local swap we also need a spy point.

Other encodings

- Similar translations for the rest of logics.
- For local swap we also need a spy point.
- Global cases and both versions of bridge are more similar to global sabotage.

Other encodings

- Similar translations for the rest of logics.
- For local swap we also need a spy point.
- Global cases and both versions of bridge are more similar to global sabotage.
- Proofs are adaptable for other versions of RCML (e.g., change adjacent edges but don't move).

Ending remarks

+ Provide undecidability results for the six RCML we investigate.

Ending remarks

+ Provide undecidability results for the six RCML we investigate.
+ Complete the picture of their computational behaviour.

Ending remarks

+ Provide undecidability results for the six RCML we investigate.
+ Complete the picture of their computational behaviour.
+ Improve previous proofs for local swap and global sabotage, avoiding redundant encodings of the tiling problem or PCP.

Ending remarks

+ Provide undecidability results for the six RCML we investigate.
+ Complete the picture of their computational behaviour.
+ Improve previous proofs for local swap and global sabotage, avoiding redundant encodings of the tiling problem or PCP.

- Finite satisfiability

Ending remarks

+ Provide undecidability results for the six RCML we investigate.
+ Complete the picture of their computational behaviour.
+ Improve previous proofs for local swap and global sabotage, avoiding redundant encodings of the tiling problem or PCP.
- Finite satisfiability
  - undecidable for multi-modal sabotage logic

Ending remarks

\+ Provide undecidability results for the six RCML we investigate.

\+ Complete the picture of their computational behaviour.

\+ Improve previous proofs for local swap and global sabotage, avoiding redundant encodings of the tiling problem or PCP.

\- Finite satisfiability

- undecidable for multi-modal sabotage logic
- decidable for mono-modal local swap/sabotage.

Ending remarks

+ Provide undecidability results for the six RCML we investigate.
+ Complete the picture of their computational behaviour.
+ Improve previous proofs for local swap and global sabotage, avoiding redundant encodings of the tiling problem or PCP.

- Finite satisfiability
  - undecidable for multi-modal sabotage logic
  - decidable for mono-modal local swap/sabotage.
- Proof systems.