Do you know if the road is still there?

Sabotage Modal Logic with Knowledge

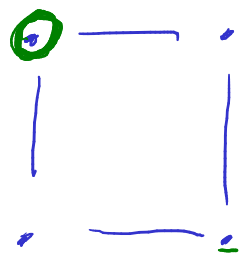
Malvin Gattinger — with Sujata Ghosh and Saptarshi Sahoo

9 April 2026, LIRa Seminar at ILLC

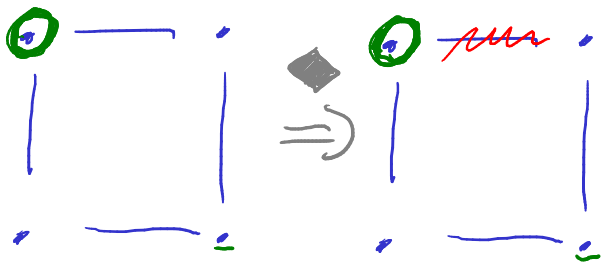
Outline

Overview

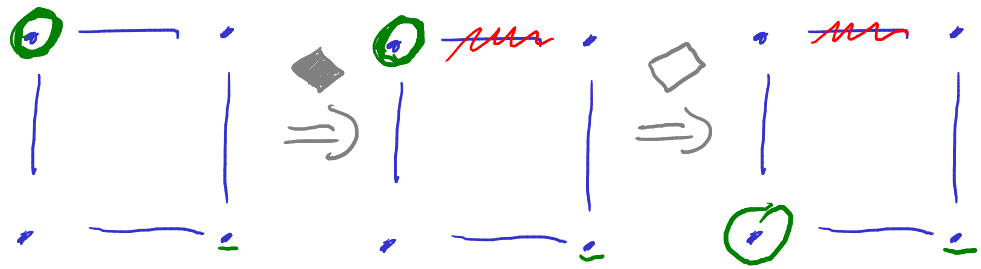
Sabotage Example



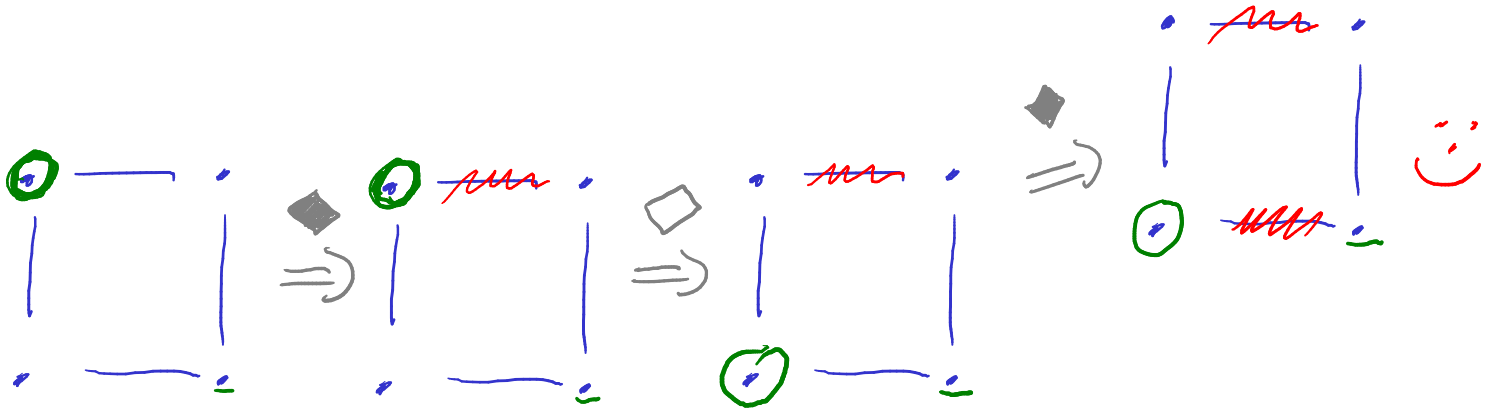
Sabotage Example



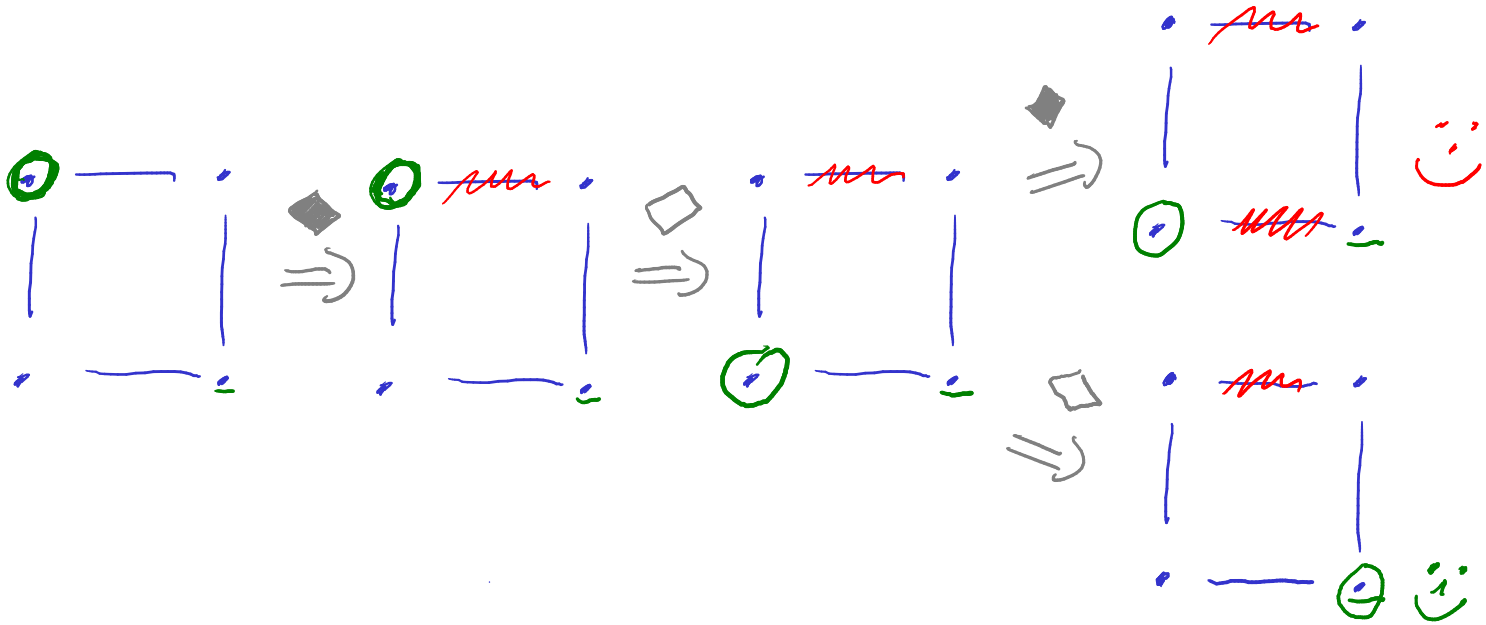
Sabotage Example



Sabotage Example



Sabotage Example



Sabotage in the real world

Practical applications of Sabotage:

- Deutsche Bahn vs Train Travelers
- Gemeente Amsterdam vs Cyclists
- Sanctions vs Air Traffic

SML Literature

- J. van Benthem (2002/5): *An Essay on Sabotage and Obstruction*
- C. Löding and P. Rohde (2003): *Model checking and satisfiability for sabotage modal logic* and *Solving the Sabotage Game Is PSPACE-Hard*
- G. Aucher, J. van Benthem, D. Grossi (2018): *Modal Logics of Sabotage Revisited*

SML Literature

- J. van Benthem (2002/5): *An Essay on Sabotage and Obstruction*
- C. Löding and P. Rohde (2003): *Model checking and satisfiability for sabotage modal logic* and *Solving the Sabotage Game Is PSPACE-Hard*
- G. Aucher, J. van Benthem, D. Grossi (2018): *Modal Logics of Sabotage Revisited*

There is a lot more. See for example the PhD theses by Philipp Rohde (2005), Lena Kurzen (2011), Raul Alberto Fervari (2013) and Lei Li (2023).

SML Syntax and Semantics

SML Syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \blacklozenge\varphi$$

SML Syntax and Semantics

SML Syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \blacklozenge\varphi$$

A *pre-model* $\mathcal{M} = (N, V)$: set of nodes N and valuation V.
A *state* $S = (R, s)$: current relation $R \subseteq N \times N$ and location $s \in N$.
Let $\mathrm{links}(S) := R$ and $\mathrm{loc}(S) := s$.

SML Syntax and Semantics

SML Syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \blacklozenge\varphi$$

A *pre-model* $\mathcal{M} = (N, V)$: set of nodes N and valuation V.
A *state* $S = (R, s)$: current relation $R \subseteq N \times N$ and location $s \in N$.
Let $\mathrm{links}(S) := R$ and $\mathrm{loc}(S) := s$.

Actions are partial functions:

- $\Diamond(x, y)$ move from x to y
- $\blacklozenge(x, y)$ delete the edge from x to y

SML Syntax and Semantics

SML Syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \blacklozenge\varphi$$

A *pre-model* $\mathcal{M} = (N, V)$: set of nodes N and valuation V.
A *state* $S = (R, s)$: current relation $R \subseteq N \times N$ and location $s \in N$.
Let $\mathrm{links}(S) := R$ and $\mathrm{loc}(S) := s$.

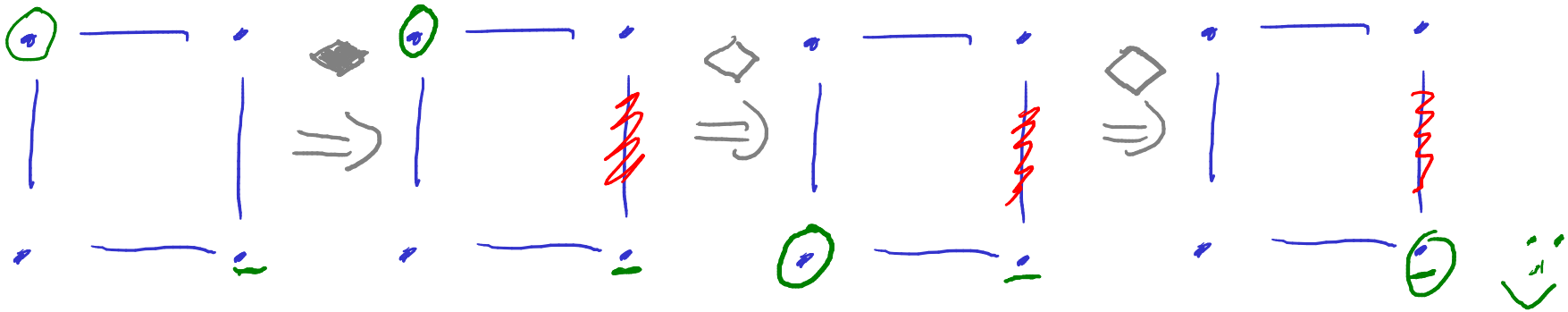
Actions are partial functions:

- $\Diamond(x, y)$ move from x to y
- $\blacklozenge(x, y)$ delete the edge from x to y

A *history* is a list of actions. Example: $\Diamond(1,2)\blacklozenge(2,3)\Diamond(2,4)$.
Given a state S, let $\mathcal{H}(S)$ be the set of all *executable* histories.

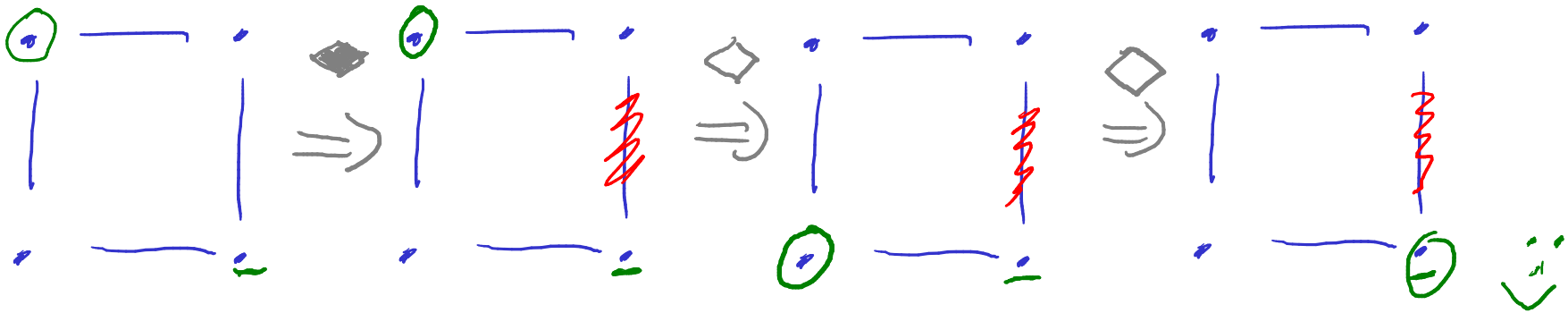
SML Semantics Example

Let g be only true at the goal.
In the left-most state we have $\blacklozenge\Diamond\Diamond g$.



SML Semantics Example

Let g be only true at the goal.
In the left-most state we have $\blacklozenge\Diamond\Diamond g$.



We also have:

- $\blacklozenge\blacklozenge\Diamond\Diamond g$
- $\blacklozenge\blacklozenge\neg\Diamond\Diamond g$
- $\blacklozenge\blacklozenge\neg\Diamond\top$
- $\blacksquare\blacksquare\blacksquare\neg\Diamond\Diamond g$

Overview

SML+K

We add $K\varphi$ to mean "traveller knows that φ":

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \blacklozenge\varphi \mid K\varphi$$

SML+K

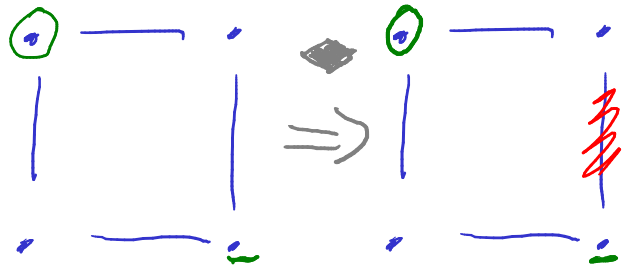
We add $K\varphi$ to mean "traveller knows that φ":

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \blacklozenge\varphi \mid K\varphi$$

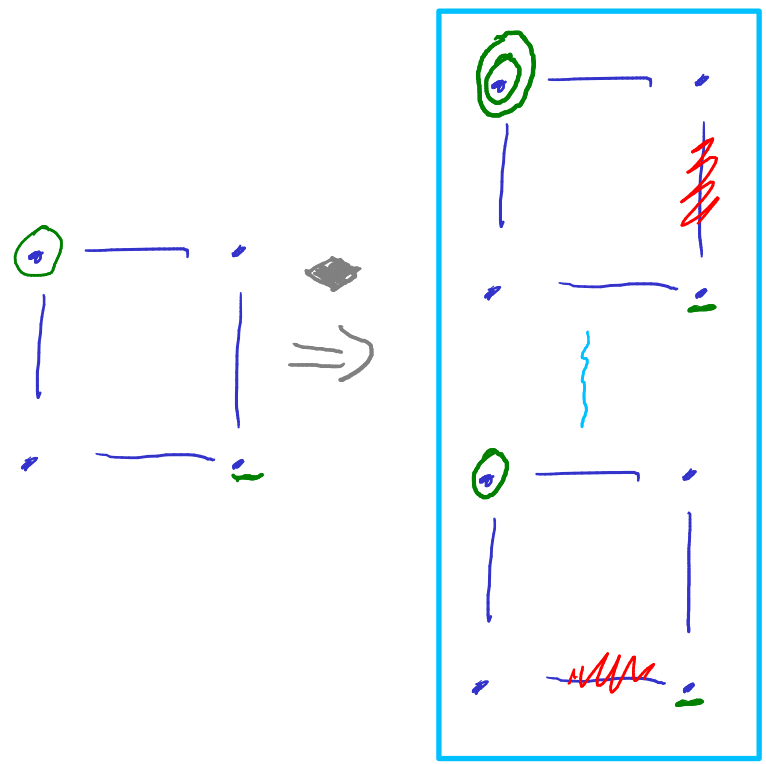
Assumptions

- The initial graph is known.
- We only model the knowledge of traveller, not demon.
- Deletions may introduce uncertainty.
- Traveller "sees" all edges at current location.

What does traveller know?



What does traveller know?



Graph Semantics: Graph States

A graph state (S, κ) for some pre-model $\mathcal{M} = (N, V)$ has:

- $S \in \mathcal{S}(\mathcal{M})$ is a state $(\mathsf{links}(S), \mathsf{loc}(S))$ of $\mathcal{M}$ and
- $\kappa \subseteq \mathcal{P}(N \times N)$ is a *set of relations* over N,

Graph Semantics: Graph States

A graph state (S, κ) for some pre-model $\mathcal{M} = (N, V)$ has:

- $S \in \mathcal{S}(\mathcal{M})$ is a state $(\mathsf{links}(S), \mathsf{loc}(S))$ of $\mathcal{M}$ and
- $\kappa \subseteq \mathcal{P}(N \times N)$ is a *set of relations* over N,

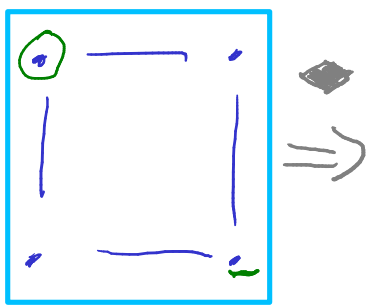
We want $\mathsf{links}(S) \in \kappa$ to ensure $K\varphi \to \varphi$.

Graph Semantics: Graph States

A graph state (S, κ) for some pre-model $\mathcal{M} = (N, V)$ has:

- $S \in \mathcal{S}(\mathcal{M})$ is a state $(\mathsf{links}(S), \mathsf{loc}(S))$ of $\mathcal{M}$ and
- $\kappa \subseteq \mathcal{P}(N \times N)$ is a *set of relations* over N,

We want $\mathsf{links}(S) \in \kappa$ to ensure $K\varphi \to \varphi$.

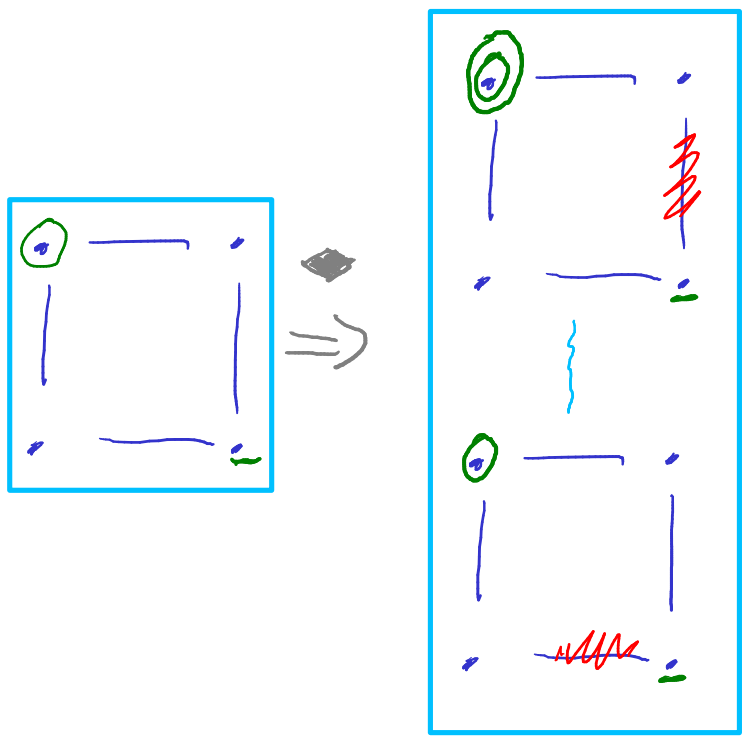


Graph Semantics: Graph States

A graph state (S, κ) for some pre-model $\mathcal{M} = (N, V)$ has:

- $S \in \mathcal{S}(\mathcal{M})$ is a state $(\mathsf{links}(S), \mathsf{loc}(S))$ of $\mathcal{M}$ and
- $\kappa \subseteq \mathcal{P}(N \times N)$ is a *set of relations* over N,

We want $\mathsf{links}(S) \in \kappa$ to ensure $K\varphi \to \varphi$.

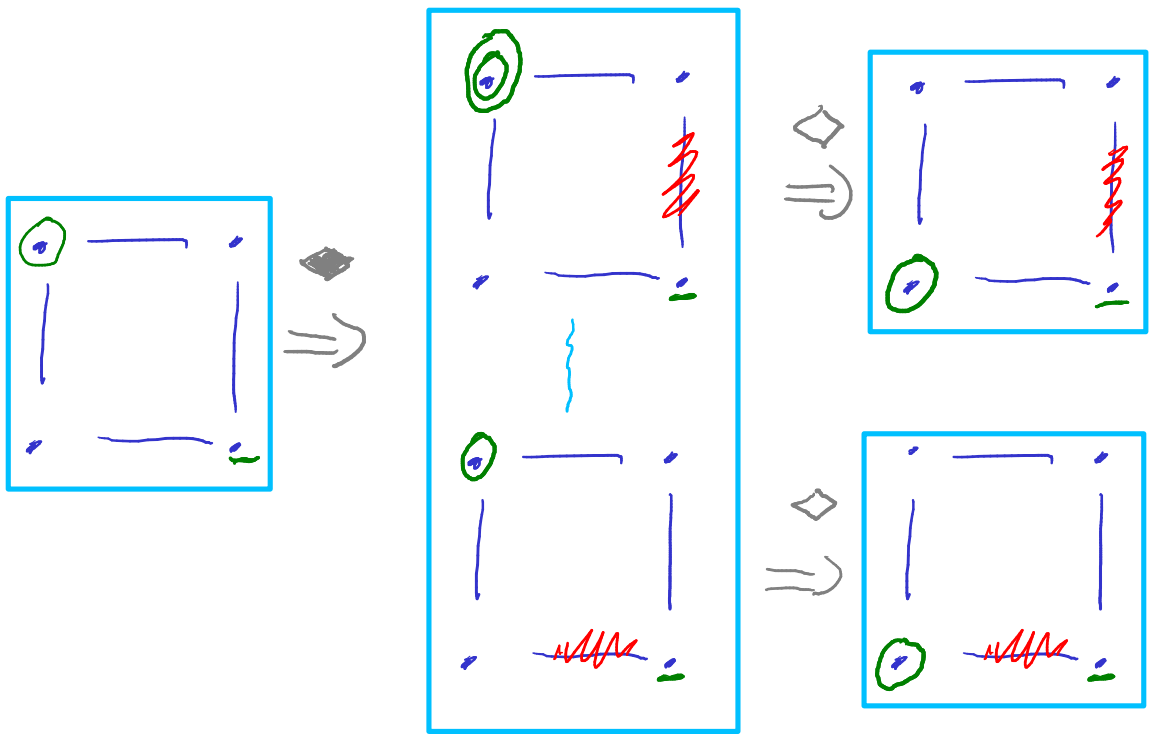


Graph Semantics: Graph States

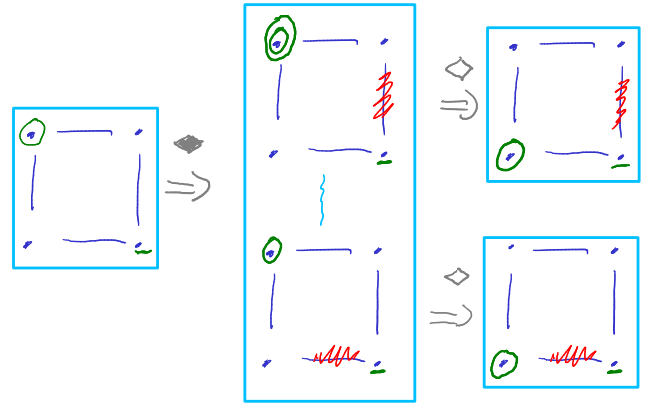
A graph state (S, κ) for some pre-model $\mathcal{M} = (N, V)$ has:

- $S \in \mathcal{S}(\mathcal{M})$ is a state $(\mathsf{links}(S), \mathsf{loc}(S))$ of $\mathcal{M}$ and
- $\kappa \subseteq \mathcal{P}(N \times N)$ is a *set of relations* over N,

We want $\mathsf{links}(S) \in \kappa$ to ensure $K\varphi \to \varphi$.



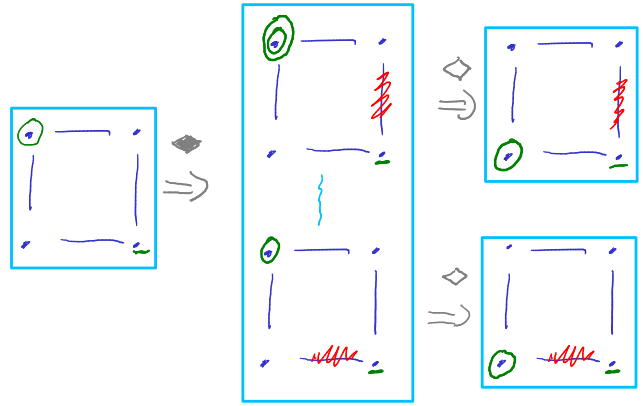
Graph Semantics: Updating κ



At any state S, what does the traveller see?

$$\mathsf{see}(S) := \{(x, y) \in \mathsf{links}(S) \mid \mathsf{loc}(S) \in \{x, y\}\}.$$

Graph Semantics: Updating κ

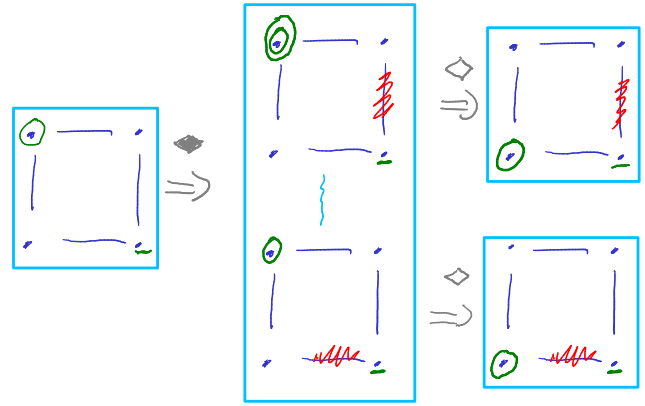


At any state S, what does the traveller see?

$$\mathsf{see}(S) := \{(x, y) \in \mathsf{links}(S) \mid \mathsf{loc}(S) \in \{x, y\}\}.$$

How do we now update a graph state (S, κ)?

Graph Semantics: Updating κ



At any state S, what does the traveller see?

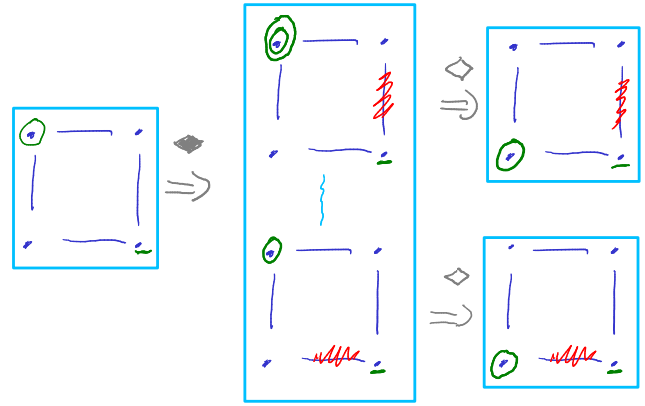
$$\mathsf{see}(S) := \{(x, y) \in \mathsf{links}(S) \mid \mathsf{loc}(S) \in \{x, y\}\}.$$

How do we now update a graph state (S, κ)?

When *making a move we learn*: $(S, \kappa)^{\Diamond(s,t)} := (S^{\Diamond(s,t)}, \kappa')$ where

$$\kappa' := \{R \in \kappa \mid \mathsf{see}(S^{\Diamond(s,t)}) = \mathsf{see}((R, t))\}$$

Graph Semantics: Updating κ



At any state S, what does the traveller see?

$$\mathsf{see}(S) := \{(x, y) \in \mathsf{links}(S) \mid \mathsf{loc}(S) \in \{x, y\}\}.$$

How do we now update a graph state (S, κ)?

When *making a move we learn*: $(S, \kappa)^{\Diamond(s,t)} := (S^{\Diamond(s,t)},\ \kappa')$ where

$$\kappa' := \{R \in \kappa \mid \mathsf{see}(S^{\Diamond(s,t)}) = \mathsf{see}((R, t))\}$$

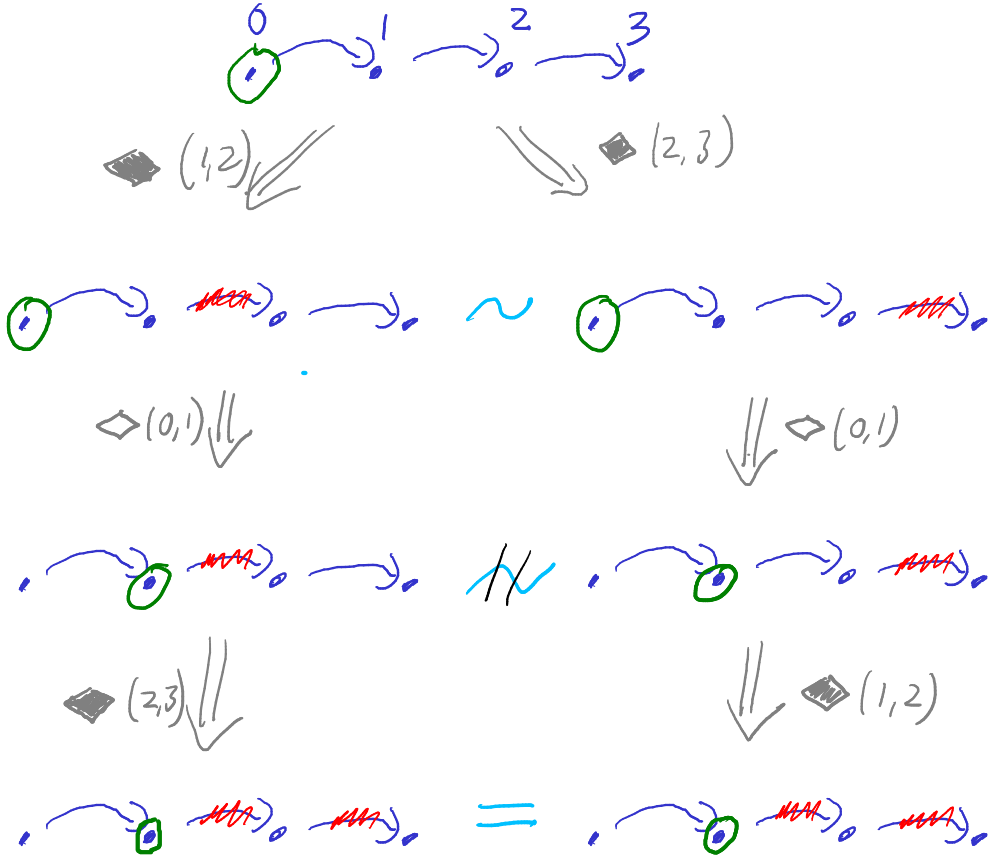
But *deletion may also create ignorance*: $(S, \kappa)^{\blacklozenge(t,u)} := (S^{\blacklozenge(t,u)}, \kappa')$ where

$$\kappa' := \left\{ R \setminus \{(v, w)\} \mid \begin{array}{l} R \in \kappa,\ (v, w) \in R,\ \text{and} \\ \mathsf{see}(S^{\blacklozenge(t,u)}) = \mathsf{see}((R \setminus \{(v, w)\},\ \mathsf{loc}(S))) \end{array} \right\}$$

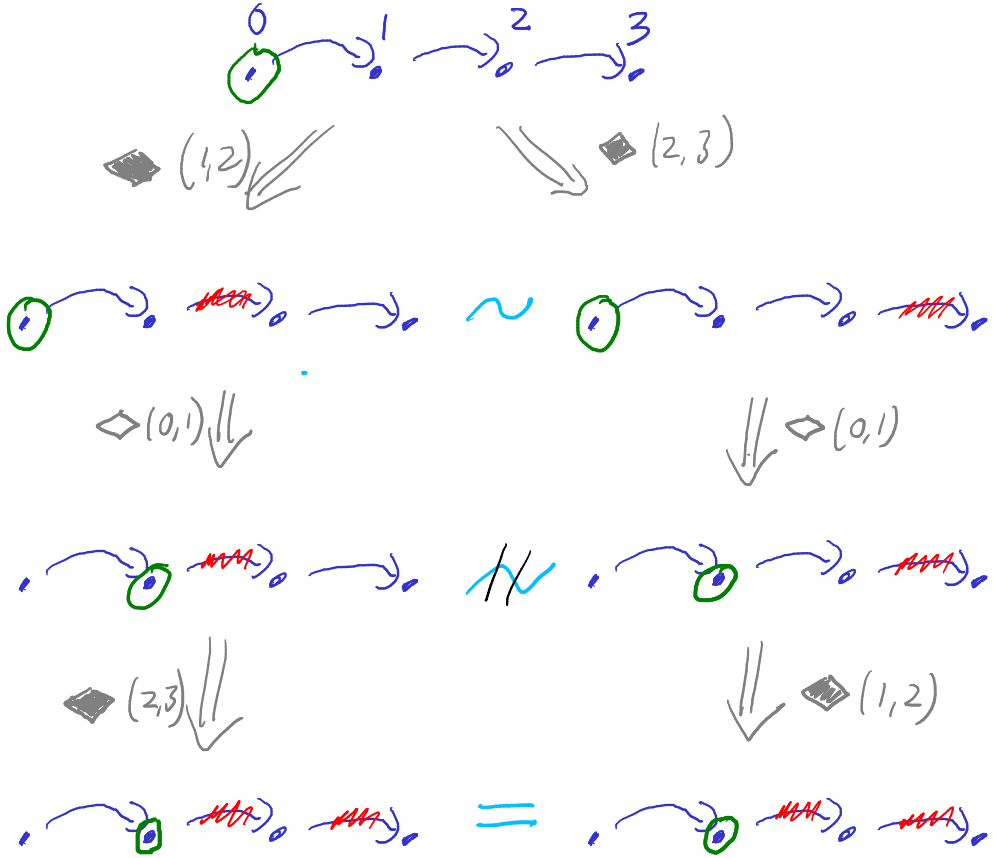
Graph Semantics: Definition

$$
\begin{array}{lll}
(S,\kappa) \models p & \text{iff} & p \in V(\mathsf{loc}(S)) \\
(S,\kappa) \models \neg\varphi & \text{iff} & \text{not } (S,\kappa) \models \varphi \\
(S,\kappa) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (S,\kappa) \models \varphi_1 \text{ and } (S,\kappa) \models \varphi_1 \\
(S,\kappa) \models \Diamond\varphi & \text{iff} & \exists t, (\mathsf{loc}(S), t) \in \mathsf{links}(S) \text{ and } (S,\kappa)^{\Diamond(\mathsf{loc}(S),t)} \models \varphi \\
(S,\kappa) \models \blacklozenge\varphi & \text{iff} & \exists (t,u) \in \mathsf{links}(S), (S,\kappa)^{\blacklozenge(t,u)} \models \varphi \\
(S,\kappa) \models K\varphi & \text{iff} & \forall R \in \kappa, ((R, \mathsf{loc}(S)),\kappa) \models \varphi
\end{array}
$$

Graph Semantics: No perfect recall?



Graph Semantics: No perfect recall?



After these two histories we arrive at the same graph state:

- $\blacklozenge(1,2)\Diamond(0,1)\blacklozenge(2,3)$
- $\blacklozenge(2,3)\Diamond(0,1)\blacklozenge(1,2)$

So traveller has no memory!? Could we falsify $K\blacksquare\varphi \to \blacksquare K\varphi$ etc?

History Semantics: History States and Epistemic Relation

Idea: instead of "sets of possible graphs now",
let traveller *consider possible histories*.

History Semantics: History States and Epistemic Relation

Idea: instead of "sets of possible graphs now",
let traveller *consider possible histories*.

A history state is a pair (S, σ) where σ is a history executable on S.

History Semantics: History States and Epistemic Relation

Idea: instead of "sets of possible graphs now",
let traveller *consider possible histories*.

A history state is a pair (S, σ) where σ is a history executable on S.

When are two histories equivalent?

$$\begin{array}{lll} \epsilon \sim \epsilon & & \text{always} \\ \sigma\Diamond(x,y) \sim \tau\Diamond(x,y) & \text{iff} & \sigma \sim \tau \text{ and } \mathsf{see}(S_0, \sigma\Diamond(x,y)) = \mathsf{see}(S_0, \tau\Diamond(x,y)) \\ \sigma\blacklozenge(x,y) \sim \tau\blacklozenge(x',y') & \text{iff} & \sigma \sim \tau \text{ and } \mathsf{see}(S_0, \sigma\blacklozenge(x,y)) = \mathsf{see}(S_0, \tau\blacklozenge(x',y')) \end{array}$$

We never consider a different type ($\Diamond \neq \blacklozenge$) possible.

History Semantics: History States and Epistemic Relation

Idea: instead of "sets of possible graphs now",
let traveller *consider possible histories*.

A history state is a pair (S, σ) where σ is a history executable on S.

When are two histories equivalent?

$$\begin{array}{lll} \epsilon \sim \epsilon & & \text{always} \\ \sigma\Diamond(x,y) \sim \tau\Diamond(x,y) & \text{iff} & \sigma \sim \tau \text{ and } \mathsf{see}(S_0, \sigma\Diamond(x,y)) = \mathsf{see}(S_0, \tau\Diamond(x,y)) \\ \sigma\blacklozenge(x,y) \sim \tau\blacklozenge(x',y') & \text{iff} & \sigma \sim \tau \text{ and } \mathsf{see}(S_0, \sigma\blacklozenge(x,y)) = \mathsf{see}(S_0, \tau\blacklozenge(x',y')) \end{array}$$

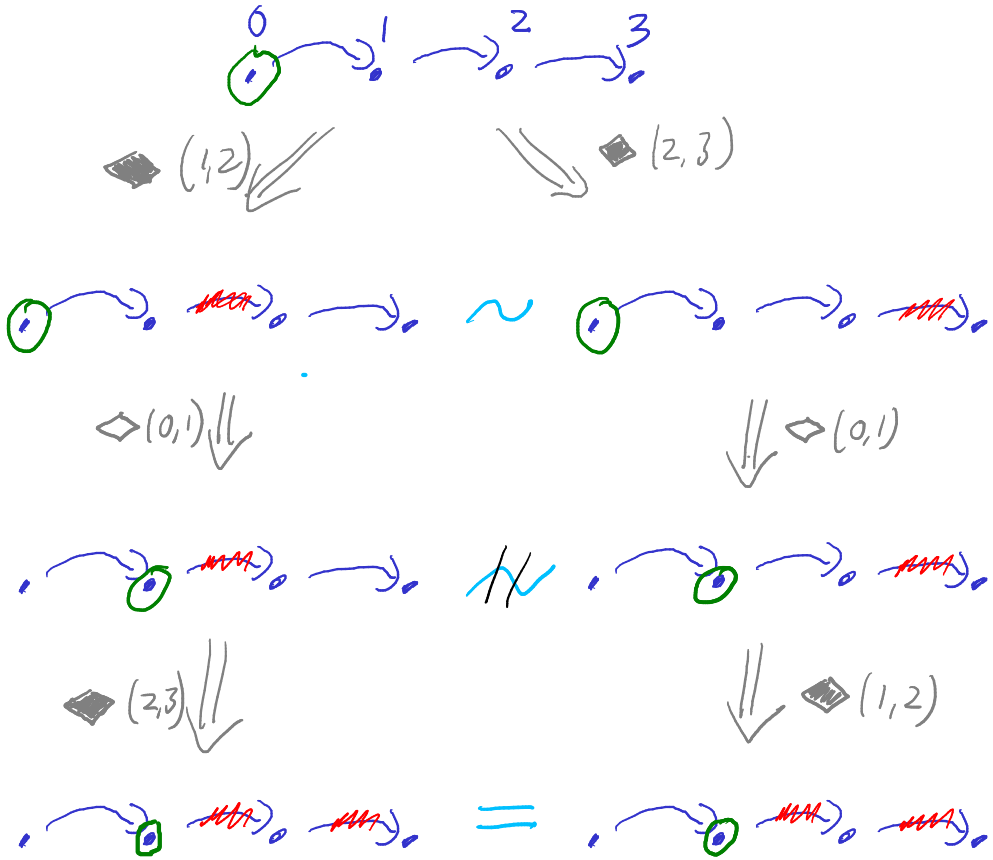
We never consider a different type ($\Diamond \neq \blacklozenge$) possible.

Side note: this is very much like call sequences in Gossip.

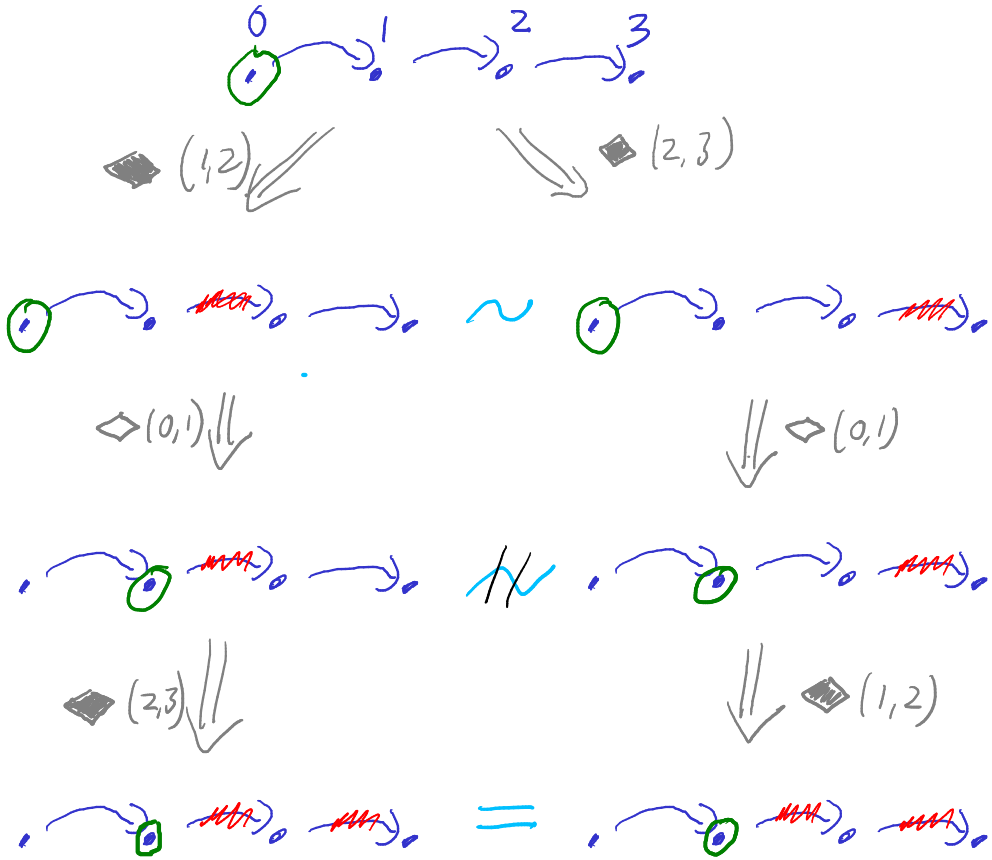
History Semantics: Definition

$$
\begin{array}{lll}
(S_0, \sigma) \models p & \text{iff} & p \in V(\mathrm{loc}((S_0, \sigma))) \\
(S_0, \sigma) \models \neg\varphi & \text{iff} & \text{not } (S_0, \sigma) \models \varphi \\
(S_0, \sigma) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (S_0, \sigma) \models \varphi_1 \text{ and } (S_0, \sigma) \models \varphi_2 \\
 & & \\
(S_0, \sigma) \models \Diamond\varphi & \text{iff} & \exists x, y \in N : \sigma\Diamond(x, y) \in \mathcal{H}(S_0) \\
 & & \text{and } (S_0, \sigma\Diamond(x, y)) \models \varphi \\
 & & \\
(S_0, \sigma) \models \blacklozenge\varphi & \text{iff} & \exists x, y \in N : \sigma\blacklozenge(x, y) \in \mathcal{H}(S_0) \\
 & & \text{and } (S_0, \sigma\blacklozenge(x, y)) \models \varphi \\
 & & \\
(S_0, \sigma) \models K\varphi & \text{iff} & \forall \tau : \tau \sim \sigma \text{ we have } (S_0, \tau) \models \varphi
\end{array}
$$

History Semantics: Example

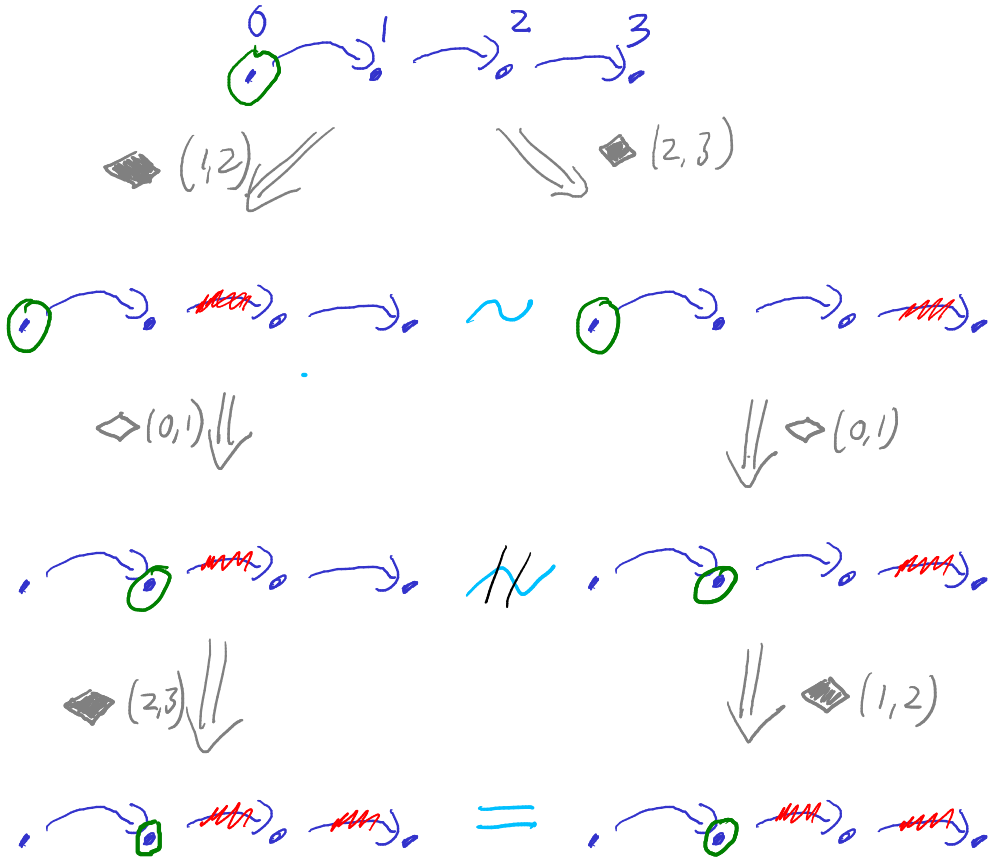


History Semantics: Example



But now we have $\blacklozenge(1,2)\Diamond(0,1)\blacklozenge(2,3) \not\sim \blacklozenge(2,3)\Diamond(0,1)\blacklozenge(1,2)$.

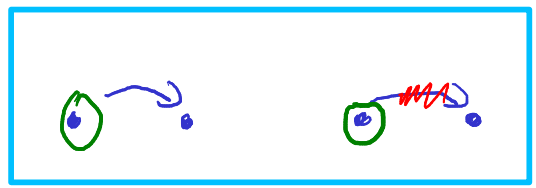
History Semantics: Example



But now we have $\blacklozenge(1,2)\Diamond(0,1)\blacklozenge(2,3) \not\sim \blacklozenge(2,3)\Diamond(0,1)\blacklozenge(1,2)$.

So the history semantics has perfect recall.
Both $K\blacksquare\varphi \to \blacksquare K\varphi$ and $K\Box\varphi \to \Box K\varphi$ are valid.

Side Note: unreachable models



Equivalence of the Semantics

A graph state (S, κ) is *reachable* iff there exist $S_0 \in \mathcal{S}(\mathcal{M})$ and $\sigma \in \mathcal{H}(S_0)$, such that $(S_0, \{\mathsf{links}(S_0)\})^\sigma = (S, \kappa)$.

Equivalence of the Semantics

A graph state (S, κ) is *reachable* iff there exist $S_0 \in \mathcal{S}(\mathcal{M})$ and $\sigma \in \mathcal{H}(S_0)$, such that $(S_0, \{\mathsf{links}(S_0)\})^\sigma = (S, \kappa)$.

Theorem

The reachable graph semantics and the history semantics coincide. That is, φ is valid on reachable graph states if and only if it is valid on history states.

Equivalence of the Semantics

A graph state (S, κ) is *reachable* iff there exist $S_0 \in \mathcal{S}(\mathcal{M})$ and $\sigma \in \mathcal{H}(S_0)$, such that $(S_0, \{\mathsf{links}(S_0)\})^\sigma = (S, \kappa)$.

Theorem

The reachable graph semantics and the history semantics coincide. That is, φ is valid on reachable graph states if and only if it is valid on history states.

Intuition: it does not matter whether traveller stores a list of

- possible graphs for right now, or
- possible histories so far.

Equivalence of the Semantics: Proof Ideas

Theorem

Any φ is valid on reachable graph states iff it is valid on history states.

Equivalence of the Semantics: Proof Ideas

Theorem

Any φ is valid on reachable graph states iff it is valid on history states.

Key Lemmas:

- "Whenever $\sigma \sim \tau$ then the κ reached by both sequences is equal."
- If (S, κ) is reached by σ from S_0, then:

$$\forall R \in \kappa \; \exists \tau \in \mathcal{H}(S_0), \; (S_0^\tau = (R, \mathsf{loc}(S_0^\tau)), \text{ and } \sigma \sim \tau$$

(Both by induction on σ.)

Equivalence of the Semantics: Proof Ideas

Theorem

Any φ is valid on reachable graph states iff it is valid on history states.

Key Lemmas:

- "Whenever $\sigma \sim \tau$ then the κ reached by both sequences is equal."
- If (S, κ) is reached by σ from S_0, then:

$$\forall R \in \kappa \; \exists \tau \in \mathcal{H}(S_0), \; (S_0^\tau = (R, \mathsf{loc}(S_0^\tau)), \text{ and } \sigma \sim \tau$$

(Both by induction on σ.)

- If (S, κ) is reached by σ from S_0, then:

$$(S_0, \sigma) \models \varphi \iff (S, \kappa) \models \varphi$$

(By induction on φ.)

Interesting Validities

- S5 axioms for K
- necessitation for K
- $K\varphi$, where φ is valid in SML

Interesting Validities

- S5 axioms for K
- necessitation for K
- $K\varphi$, where φ is valid in SML

- $\Diamond\top \to \blacklozenge\top$ ("if a move is possible, then a deletion is possible")
- $\Box\bot \to K\Box\bot$ ("if no move is possible, then traveller knows that")
- $\Box\bot \to \blacksquare\Box\bot$ ("if no move is possible, deleting will not change it")

Interesting Validities

- S5 axioms for K
- necessitation for K
- $K\varphi$, where φ is valid in SML

- $\Diamond\top \to \blacklozenge\top$ ("if a move is possible, then a deletion is possible")
- $\Box\bot \to K\Box\bot$ ("if no move is possible, then traveller knows that")
- $\Box\bot \to \blacksquare\Box\bot$ ("if no move is possible, deleting will not change it")

- $K\Box\varphi \to \Box K\varphi$ (perfect recall)
- $K\blacksquare\varphi \to \blacksquare K\varphi$ (perfect recall)
- $\Box p \to \blacklozenge K\Box p \wedge \blacksquare K\Box p$

Overview

Lean Defs

```
/-- A history is a list of actions. NOTE: the head is the newest/last action.
So σ = a₁ a₂...aₙ in the paper is represented as [aₙ,.., a₂, a₁] here in Lean. -/
def History {N} := List (@Action N)

/-- Apply a single ‘Action‘ to a ‘State‘. -/
def State.apply {M : @PreModel P} : (Es : State M) → @Action M.N → Option (State M)
  | ⟨E, s⟩, .move x y =>
      have := M.N_decEq
      if x = s ∧ (x,y) ∈ E then .some ⟨E, y⟩ else none
  | ⟨E, s⟩, .del  x y =>
      have := M.N_decEq
      if (x,y) ∈ E then .some ⟨E.erase (x,y), s⟩ else none

/-- Execute history on a ‘State‘.
Partial function because not all ‘History‘ values are executable. -/
def State.after {M : @PreModel P} : State M → @History M.N → Option (State M)
  | S, [] => .some S
  | S, (a :: rest) => after S rest >>= fun T => T.apply a

def History.isExectuable {M : @PreModel P} (S : State M) (σ : @History M.N) : Bool :=
  (S.after σ).isSome
```

Lean Live Demo :-)

Wish me luck . . .

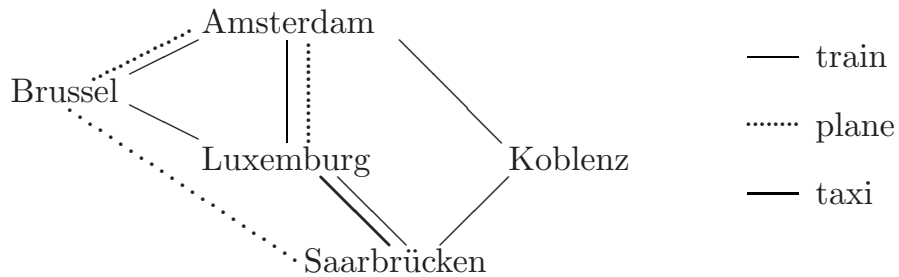
Lean Example

```
/-- A line. -/
def pRel : Relation (Fin 4) := [(0,1),(1,2),(2,3)]

/-- Initial state with `0` as starting location. -/
def pS₀ : State pModel := ⟨pRel, (0 : Fin 4)⟩

-- Now consider these two histories: (Note that we list histories backwards in Lean).
-- They lead to the same `State`:
#eval pS₀.after [ .del 2 3, .move 0 1, .del 1 2 ] -- some ([(0,1)], 1)
#eval pS₀.after [ .del 1 2, .move 0 1, .del 2 3 ] -- some ([(0,1)], 1)

-- But after the second action (move 0 1) traveller sees different edges:
#eval HState.see (⟨pS₀, [.move 0 1, .del 1 2], by decide⟩) -- [(0, 1)]
#eval HState.see (⟨pS₀, [.move 0 1, .del 2 3], by decide⟩) -- [(0,1), (1,2)]
```

Overview

Future Work

- More or less knowledge / observation
- Multi-graph to be more realistic / give traveller higher chances?
  Actually, van Benthem (2002/5) used multigraphs:



- MC Complexity — as simple/hard as SML without K already?
- Bisimulation and Standard Translation (as done by Aucher et al. 2018)
- Embed/simulate it all in Standard DEL
  Tricky: embedding is for a fixed number or vertices!
- Lean: finish equivalence proof

Conclusion

Sabotage Modal Logic + Knowledge

- We can extend SML with K
- Two semantics
  - Graph based
  - History based

  They turn out to be equivalent!
- Model Checking in Lean

Future Work:

- Variants, Bisimulation, Benchmarks, ...