Epistemic Logic for Security

Lorenzo Ceragioli

January 9, 2018

Table of Contents

Epistemic Logic

Hypothesis (Safe approximations)

- Doxastic Logic, logic of belief
  V.S
  Epistemic Logic, logic of knowledge
- Everything agents knows is true
- Logical Omniscience
  - perfect reasoner
  - no awareness problems

Language

Given

At	set of atomic propositions
Ag	set of agent symbols
Op	set of modal operators

where usually *Op* depends on *Ag*.

$\varphi \in$ ***L(At, Op, Ag)*** defined by *BNF*

$$\varphi \; = \; p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi$$

where $p \in$ *At* and $\Box \in$ *Op*.

Language

Given

At	set of atomic propositions
Ag	set of agent symbols
Op	set of modal operators

where usually *Op* depends on *Ag*.

$\varphi \in$ ***L(At, Op, Ag)*** defined by *BNF*

$$\varphi \;=\; p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi$$

where $p \in$ *At* and $\Box \in$ *Op*.

Most simple language: ***Op*** $= \{K_a \mid a \in \mathit{Ag}\}$

Only Simple Knowledge Language

$$\begin{aligned} \mathsf{At} &= \{S_a \mid a \in \mathit{Ag} \wedge S \in \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}\} \\ \mathsf{Ag} &= \{A, B, C\} \text{ for Alice, Bob and Charlie} \\ \mathsf{Op} &= \{K_a \mid a \in \mathit{Ag}\} \end{aligned}$$

In ***L(At, Op , Ag)*** we can say

$\heartsuit_B$	Bob has a hearth card
$K_A \spadesuit_A$	Alice knows that she has a spade card
$\neg K_B \heartsuit_B$	Bob doesn't know he has a hearth card
$\neg K_B K_A \spadesuit_A$	Bob doesn't know that Alice knows his card is spade
$\diamondsuit_C \wedge K_C \clubsuit_C$	Charlie has a diamond card but knows (know, not believe!) to have a club one

Semantics

Given At and Ag we define a ***Kripke model M*** as $\boldsymbol{M = (W, R, V)}$ where

Semantics

Given At and Ag we define a ***Kripke model M*** as ***M = (W, R, V)*** where

$W \neq \emptyset$ is a set of possible worlds

Semantics

Given At and Ag we define a ***Kripke model M*** as ***M = (W, R, V)*** where

W $\neq \emptyset$ is a set of possible worlds

R : $Ag \rightarrow W \times W$ is a function yielding an accessibility relation R_a for each agent a, ideally the world w' is accessible from w using R_a (we will write $w \xrightarrow{a} w'$) if in world w the agent a think w' to be possible given its knowledge

Semantics

Given *At* and *Ag* we define a ***Kripke model M*** as ***M = (W, R, V)*** where

W $\neq \emptyset$ is a set of possible worlds

R $: Ag \to W \times W$ is a function yielding an accessibility relation R_a for each agent a, ideally the world w' is accessible from w using R_a (we will write $w \xrightarrow{a} w'$) if in world w the agent a think w' to be possible given its knowledge

V $: W \to (At \to \{true, false\})$ is a function that for each world w yield a propositional valuation $V(w)$ such that $V(w)(p) \mapsto true$ iff p is true in world w

Truth in a Kripke Model

Given a Kripke Model $M = (W,R,V)$ and a world w we define what it means for a formula φ to be true in (M, w), written $M, w \vDash \varphi$

$M, w \vDash p$	iff $V(w)(p) = true$ where $p \in At$
$M, w \vDash \varphi \wedge \psi$	iff $M, w \vDash \varphi$ and $M, w \vDash \psi$
$M, w \vDash \varphi \vee \psi$	iff $M, w \vDash \varphi$ or $M, w \vDash \psi$
$M, w \vDash \neg\varphi$	iff $M, w \nvDash \varphi$
$M, w \vDash K_a\varphi$	iff $M, w' \vDash \varphi$ for all w' such that $(w, w') \in R_a$

Truth in a Kripke Model

Given a Kripke Model $M = (W,R,V)$ and a world w we define what it means for a formula φ to be true in $(M,\, w)$, written $M, w \vDash \varphi$

$M, w \vDash p$	iff $V(w)(p) = true$ where $p \in At$
$M, w \vDash \varphi \wedge \psi$	iff $M, w \vDash \varphi$ and $M, w \vDash \psi$
$M, w \vDash \varphi \vee \psi$	iff $M, w \vDash \varphi$ or $M, w \vDash \psi$
$M, w \vDash \neg\varphi$	iff $M, w \nvDash \varphi$
$M, w \vDash K_a\varphi$	iff $M, w' \vDash \varphi$ for all w' such that $(w, w') \in R_a$

And we write

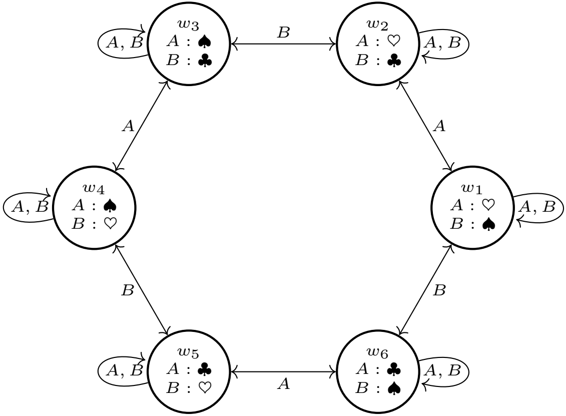
$$M \vDash \varphi \qquad \text{iff } M, w \vDash \varphi \text{ for all } w \in W$$

Example

Suppose to have two players Alice and Bob with a deck of three cards $\heartsuit$, $\spadesuit$, $\clubsuit$. Each player pick a card, each player knows only his card and the rules of the game.

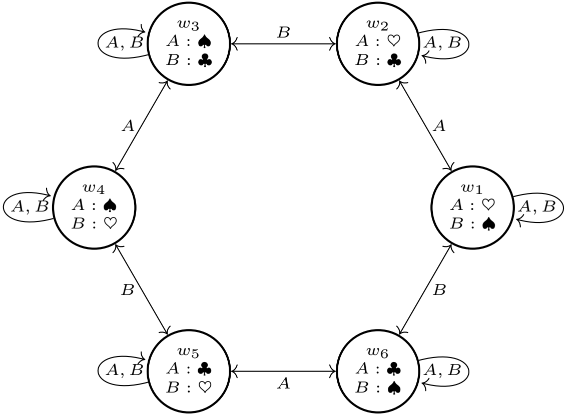
Example

Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.



Example

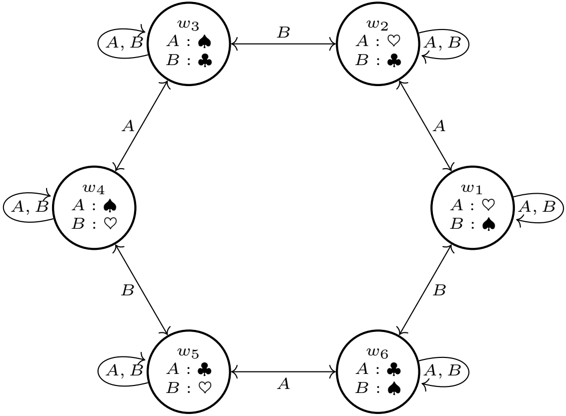
Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.



$$M, w_1 \vDash \heartsuit_A \wedge K_A \heartsuit_A$$

Example

Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.

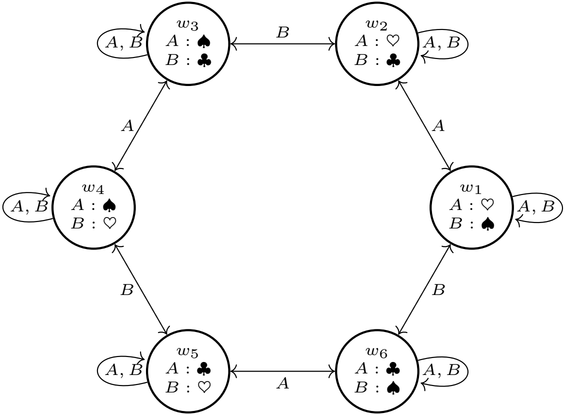


$M, w_1 \models ♡_A \land K_A ♡_A$

$M, w_1 \models \neg K_A ♠_B$

Example

Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.



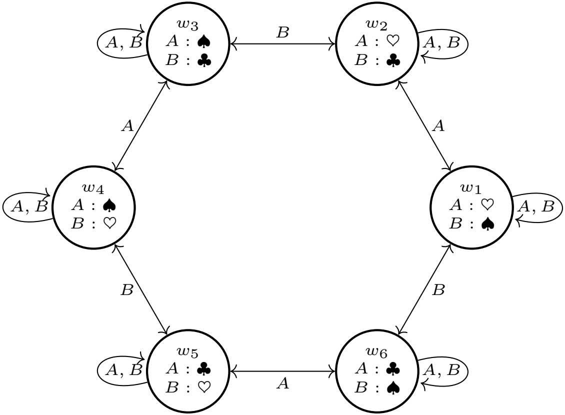
$M, w_1 \models ♡_A \wedge K_A ♡_A$

$M, w_1 \models \neg K_A ♠_B$

$M, w_1 \models K_A(♠_B \vee ♣_B)$

Example

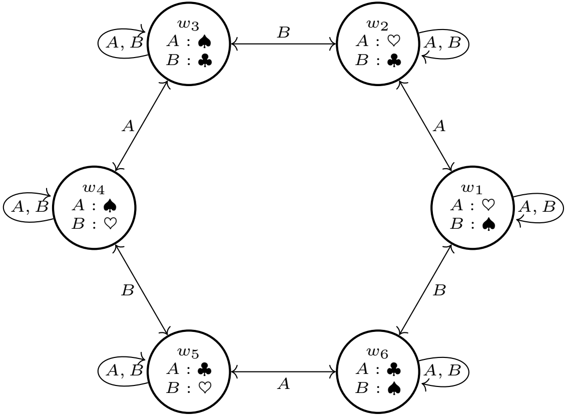
Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.



$M, w_1 \vDash \heartsuit_A \wedge K_A \heartsuit_A$
$M, w_1 \vDash \neg K_A \spadesuit_B$
$M, w_1 \vDash K_A(\spadesuit_B \vee \clubsuit_B)$
$M, w_1 \vDash K_A(\spadesuit_B \Rightarrow K_B \spadesuit_B)$

Example

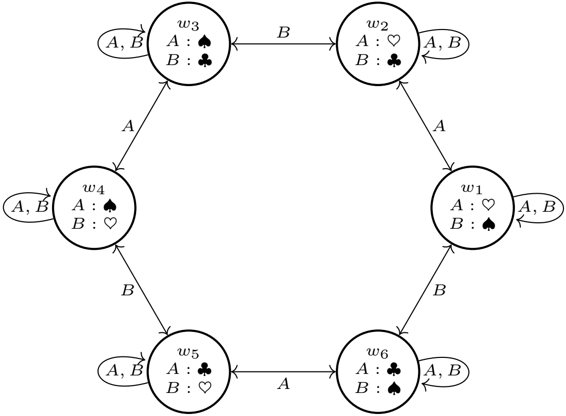
Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.



$M, w_1 \models \heartsuit_A \wedge K_A \heartsuit_A$
$M, w_1 \models \neg K_A \spadesuit_B$
$M, w_1 \models K_A(\spadesuit_B \vee \clubsuit_B)$
$M, w_1 \models K_A(\spadesuit_B \Rightarrow K_B \spadesuit_B)$
$M, w_1 \models K_A(\neg K_B \heartsuit_A)$

Example

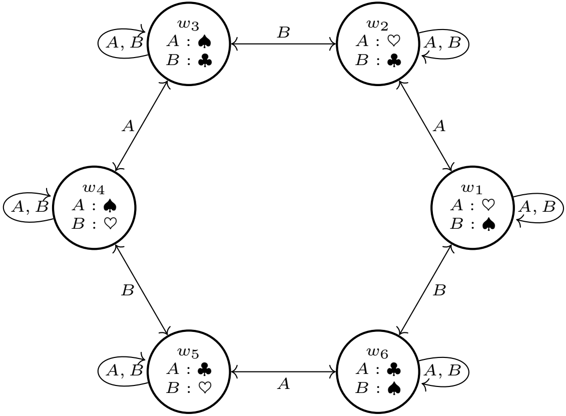
Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.



$M, w_1 \models \heartsuit_A \wedge K_A \heartsuit_A$
$M, w_1 \models \neg K_A \spadesuit_B$
$M, w_1 \models K_A(\spadesuit_B \vee \clubsuit_B)$
$M, w_1 \models K_A(\spadesuit_B \Rightarrow K_B \spadesuit_B)$
$M, w_1 \models K_A(\neg K_B \heartsuit_A)$
$M, w_1 \models K_A K_B \neg K_A \heartsuit_B)$
$M, w_1 \models K_A K_B \neg K_A \spadesuit_B)$
$M, w_1 \models K_A K_B \neg K_A \clubsuit_B)$

Example

Suppose to have two players Alice and Bob with a deck of three cards ♡, ♠, ♣. Each player pick a card, each player knows only his card and the rules of the game.



$M, w_1 \vDash \heartsuit_A \wedge K_A \heartsuit_A$
$M, w_1 \vDash \neg K_A \spadesuit_B$
$M, w_1 \vDash K_A(\spadesuit_B \vee \clubsuit_B)$
$M, w_1 \vDash K_A(\spadesuit_B \Rightarrow K_B \spadesuit_B)$
$M, w_1 \vDash K_A(\neg K_B \heartsuit_A)$
$M, w_1 \vDash K_A K_B \neg K_A \heartsuit_B)$
$M, w_1 \vDash K_A K_B \neg K_A \spadesuit_B)$
$M, w_1 \vDash K_A K_B \neg K_A \clubsuit_B)$
$M \vDash \clubsuit_B \Rightarrow K_B \clubsuit_B$

Problems with Kripke models

- Generic Kripke models ($\mathcal{K}$) are more general than what we need

Problems with Kripke models

- Generic Kripke models ($\mathcal{K}$) are more general than what we need
- They can model situations not coherent with our notion of knowledge (such as belief e.g.)

Problems with Kripke models

- Generic Kripke models ($\mathcal{K}$) are more general than what we need
- They can model situations not coherent with our notion of knowledge (such as belief e.g.)
- We need some restrictions

Problems with Kripke models

- Generic Kripke models ($\mathcal{K}$) are more general than what we need
- They can model situations not coherent with our notion of knowledge (such as belief e.g.)
- We need some restrictions
- We need restrictions upon accessibility relations

Problems with Kripke models

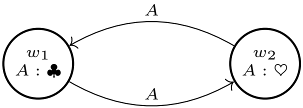
- Generic Kripke models ($\mathcal{K}$) are more general than what we need
- They can model situations not coherent with our notion of knowledge (such as belief e.g.)
- We need some restrictions
- We need restrictions upon accessibility relations
- Classes of models

Example

A very strange situation that we can model through Kripke model

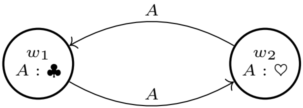
Example

A very strange situation that we can model through Kripke model



Example

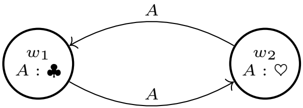
A very strange situation that we can model through Kripke model



$M, w_1 \models \clubsuit_A \wedge K_A \heartsuit_A$
Then probably it is not knowledge but belief

Example

A very strange situation that we can model through Kripke model



$M, w_1 \vDash \clubsuit_A \wedge K_A \heartsuit_A$
Then probably it is not knowledge but belief

$M, w_1 \vDash K_A \heartsuit_A \wedge \neg K_A K_A \heartsuit_A$
$M, w_1 \vDash K_A \heartsuit_A \wedge K_A K_A \clubsuit_A$
Not coherent even as belief

Kripke models of knowledge

- Knowledge is about correct information about the world

Kripke models of knowledge

- Knowledge is about correct information about the world
  $\Rightarrow R_a$ **must be reflexive**

Kripke models of knowledge

- Knowledge is about correct information about the world
  $\Rightarrow R_a$ **must be reflexive**

- $(w, w') \in R_a$ if when a is in world w thinks it is possible for world w' to be the real world, this can be if and only if w and w' are not distinguishable with a's knowledge.

Kripke models of knowledge

- Knowledge is about correct information about the world
  $\Rightarrow R_a$ **must be reflexive**

- $(w, w') \in R_a$ if when a is in world w thinks it is possible for world w' to be the real world, this can be if and only if w and w' are not distinguishable with a's knowledge.
  R_a must be coherent

Kripke models of knowledge

- Knowledge is about correct information about the world
  $\Rightarrow R_a$ **must be reflexive**

- $(w, w') \in R_a$ if when a is in world w thinks it is possible for world w' to be the real world, this can be if and only if w and w' are not distinguishable with a's knowledge.
  R_a must be coherent
  $\Rightarrow R_a$ **must be symmetric**

Kripke models of knowledge

- Knowledge is about correct information about the world
  $\Rightarrow R_a$ **must be reflexive**

- $(w, w') \in R_a$ if when a is in world w thinks it is possible for world w' to be the real world, this can be if and only if w and w' are not distinguishable with a's knowledge.
  R_a must be coherent
  $\Rightarrow R_a$ **must be symmetric**
  $\Rightarrow R_a$ **must be transitive**

Kripke models of knowledge

- Knowledge is about correct information about the world
  $\Rightarrow R_a$ **must be reflexive**

- $(w, w') \in R_a$ if when a is in world w thinks it is possible for world w' to be the real world, this can be if and only if w and w' are not distinguishable with a's knowledge.
  R_a must be coherent
  $\Rightarrow R_a$ **must be symmetric**
  $\Rightarrow R_a$ **must be transitive**

- R_a **must be an equivalence relation**

Kripke models of knowledge

- Knowledge is about correct information about the world
  $\Rightarrow R_a$ **must be reflexive**

- $(w, w') \in R_a$ if when a is in world w thinks it is possible for world w' to be the real world, this can be if and only if w and w' are not distinguishable with a's knowledge.
  R_a must be coherent
  $\Rightarrow R_a$ **must be symmetric**
  $\Rightarrow R_a$ **must be transitive**

- R_a **must be an equivalence relation**

- $\mathcal{S}5 \subseteq \mathcal{K}$ class of models

Knowledge property

Knowledge property

modus ponens: If $\mathcal{S}5 \vDash \varphi \rightarrow \psi$ and $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash \psi$

Knowledge property

modus ponens: If $\mathcal{S}5 \vDash \varphi \rightarrow \psi$ and $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash \psi$

propositional logic subsumption: If α is a substitution instance of a propositional tautology then $\mathcal{S}5 \vDash \alpha$

Knowledge property

modus ponens: If $\mathcal{S}5 \vDash \varphi \to \psi$ and $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash \psi$

propositional logic subsumption: If α is a substitution instance of a propositional tautology then $\mathcal{S}5 \vDash \alpha$

agents know logic (necessitation): If $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash K_a\varphi$

Knowledge property

modus ponens: If $\mathcal{S}5 \vDash \varphi \rightarrow \psi$ and $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash \psi$

propositional logic subsumption: If α is a substitution instance of a propositional tautology then $\mathcal{S}5 \vDash \alpha$

agents know logic (necessitation): If $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash K_a\varphi$

modus ponens on knowledge: If $\mathcal{S}5 \vDash K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)$

Knowledge property

modus ponens: If $\mathcal{S}5 \vDash \varphi \to \psi$ and $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash \psi$

propositional logic subsumption: If α is a substitution instance of a propositional tautology then $\mathcal{S}5 \vDash \alpha$

agents know logic (necessitation): If $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash K_a\varphi$

modus ponens on knowledge: If $\mathcal{S}5 \vDash K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$

knowledge internal coherence: $\mathcal{S}5 \vDash K_a\varphi \to \neg K_a\neg\varphi$

Knowledge property

modus ponens: If $\mathcal{S}5 \vDash \varphi \rightarrow \psi$ and $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash \psi$

propositional logic subsumption: If α is a substitution instance of a propositional tautology then $\mathcal{S}5 \vDash \alpha$

agents know logic (necessitation): If $\mathcal{S}5 \vDash \varphi$ then $\mathcal{S}5 \vDash K_a\varphi$

modus ponens on knowledge: If $\mathcal{S}5 \vDash K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)$

knowledge internal coherence: $\mathcal{S}5 \vDash K_a\varphi \rightarrow \neg K_a \neg\varphi$

positive introspection: $\mathcal{S}5 \vDash K_a\varphi \rightarrow K_a K_a\varphi$

Knowledge property

modus ponens: If $\mathcal{S}5 \models \varphi \rightarrow \psi$ and $\mathcal{S}5 \models \varphi$ then $\mathcal{S}5 \models \psi$

propositional logic subsumption: If α is a substitution instance of a propositional tautology then $\mathcal{S}5 \models \alpha$

agents know logic (necessitation): If $\mathcal{S}5 \models \varphi$ then $\mathcal{S}5 \models K_a\varphi$

modus ponens on knowledge: If $\mathcal{S}5 \models K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)$

knowledge internal coherence: $\mathcal{S}5 \models K_a\varphi \rightarrow \neg K_a \neg\varphi$

positive introspection: $\mathcal{S}5 \models K_a\varphi \rightarrow K_a K_a\varphi$

negative introspection: $\mathcal{S}5 \models \neg K_a\varphi \rightarrow K_a \neg K_a\varphi$

Not as simple as you could believe now

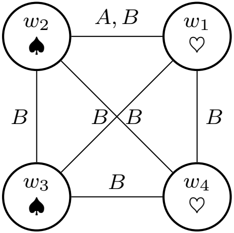
- It's not always the case that there is only one possible world for each combination of truth value for atomic predicates
- This epistemic logic is propositional, but we can consider a predicative version (also first order)
- There can be an infinite number of possible worlds (especially in the predicative case)

Example

We have a deck with two cards $\heartsuit$ and $\spadesuit$, we take one card and put it covered on the table. We have two players, Alice and Bob, Alice can cheat and look at the card, Bob cannot. Bob knows that Alice can cheat, but if she does he wouldn't know.

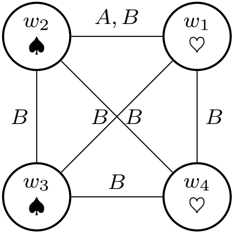
Example

We have a deck with two cards ♡ and ♠, we take one card and put it covered on the table. We have two players, Alice and Bob, Alice can cheat and look at the card, Bob cannot. Bob knows that Alice can cheat, but if she does he wouldn't know.



Example

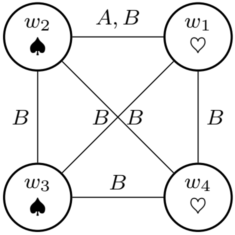
We have a deck with two cards ♡ and ♠, we take one card and put it covered on the table. We have two players, Alice and Bob, Alice can cheat and look at the card, Bob cannot. Bob knows that Alice can cheat, but if she does he wouldn't know.



$$M \vDash \neg K_B \neg K_A \spadesuit$$

Example

We have a deck with two cards ♡ and ♠, we take one card and put it covered on the table. We have two players, Alice and Bob, Alice can cheat and look at the card, Bob cannot. Bob knows that Alice can cheat, but if she does he wouldn't know.

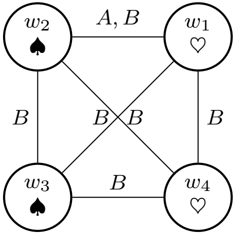


$$M \vDash \neg K_B \neg K_A ♠$$

$$M \vDash \neg K_B \neg K_A ♡$$

Example

We have a deck with two cards ♡ and ♠, we take one card and put it covered on the table. We have two players, Alice and Bob, Alice can cheat and look at the card, Bob cannot. Bob knows that Alice can cheat, but if she does he wouldn't know.



$$M \vDash \neg K_B \neg K_A ♠$$

$$M \vDash \neg K_B \neg K_A ♡$$

$$M \vDash \neg K_B (K_A ♡ \vee K_A ♠)$$

Beyond Simple Knowledge: Group Knowledge

Language extensions:

Beyond Simple Knowledge: Group Knowledge

Language extensions:

- Everyone in $G \subseteq Ag$ knows φ, we write $E_G\varphi$

Beyond Simple Knowledge: Group Knowledge

Language extensions:

- Everyone in $G \subseteq Ag$ knows φ, we write $E_G\varphi$
- Distributed Knowledge of φ among $G \subseteq Ag$, we write $D_G\varphi$

Beyond Simple Knowledge: Group Knowledge

Language extensions:

- Everyone in $G \subseteq Ag$ knows φ, we write $E_G\varphi$
- Distributed Knowledge of φ among $G \subseteq Ag$, we write $D_G\varphi$
- Common Knowledge of φ among $G \subseteq Ag$, we write $C_G\varphi$

Semantics of Group Knowledge

$M, w \vDash E_A\varphi$ iff for all w' such that $(w, w') \in R_{E_A}$, we have $M, w' \vDash \varphi$

where $R_{E_A} = \bigcup_{a \in A} R_a$

Semantics of Group Knowledge

$$M, w \vDash E_A\varphi \quad \text{iff for all } w' \text{ such that } (w, w') \in R_{E_A}, \text{ we have } M, w' \vDash \varphi$$

$$\text{where } R_{E_A} = \bigcup_{a \in A} R_a$$

$$M, w \vDash D_A\varphi \quad \text{iff for all } w' \text{ such that } (w, w') \in R_{D_A}, \text{ we have } M, w' \vDash \varphi$$

$$\text{where } R_{D_A} = \bigcap_{a \in A} R_a$$

Semantics of Group Knowledge

$M, w \vDash E_A\varphi$ iff for all w' such that $(w, w') \in R_{E_A}$, we have $M, w' \vDash \varphi$
where $R_{E_A} = \bigcup_{a \in A} R_a$

$M, w \vDash D_A\varphi$ iff for all w' such that $(w, w') \in R_{D_A}$, we have $M, w' \vDash \varphi$
where $R_{D_A} = \bigcap_{a \in A} R_a$

$M, w \vDash C_A\varphi$ iff for all w' such that $(w, w') \in R_{C_A}$, we have $M, w' \vDash \varphi$
where $R_{C_A} = (\bigcup_{a \in A} R_a)^+$

Property of Group Knowledge

Trivially

Property of Group Knowledge

Trivially

- $\mathcal{K} \vDash C_G\varphi \to E_G\varphi$

Property of Group Knowledge

Trivially

- $\mathcal{K} \vDash C_G\varphi \to E_G\varphi$
- $\mathcal{K} \vDash E_G\varphi \to K_a\varphi$ where $a \in G$

Property of Group Knowledge

Trivially

- $\mathcal{K} \models C_G\varphi \rightarrow E_G\varphi$
- $\mathcal{K} \models E_G\varphi \rightarrow K_a\varphi$ where $a \in G$
- $\mathcal{K} \models K_a\varphi \rightarrow D_G\varphi$ where $a \in G$

Property of Group Knowledge

Trivially

- $\mathcal{K} \models C_G\varphi \to E_G\varphi$
- $\mathcal{K} \models E_G\varphi \to K_a\varphi$ where $a \in G$
- $\mathcal{K} \models K_a\varphi \to D_G\varphi$ where $a \in G$
- $\mathcal{K} \models C_G\varphi \to E_G E_G\varphi$

Property of Group Knowledge

Trivially

- $\mathcal{K} \vDash C_G\varphi \to E_G\varphi$
- $\mathcal{K} \vDash E_G\varphi \to K_a\varphi$ where $a \in G$
- $\mathcal{K} \vDash K_a\varphi \to D_G\varphi$ where $a \in G$
- $\mathcal{K} \vDash C_G\varphi \to E_GE_G\varphi$
  $\mathcal{K} \vDash C_G\varphi \to E_GE_GE_G\varphi$

Property of Group Knowledge

Trivially

- $\mathcal{K} \models C_G\varphi \rightarrow E_G\varphi$
- $\mathcal{K} \models E_G\varphi \rightarrow K_a\varphi$ where $a \in G$
- $\mathcal{K} \models K_a\varphi \rightarrow D_G\varphi$ where $a \in G$
- $\mathcal{K} \models C_G\varphi \rightarrow E_GE_G\varphi$
  $\mathcal{K} \models C_G\varphi \rightarrow E_GE_GE_G\varphi$
  $\ldots$

Property of Group Knowledge

Trivially

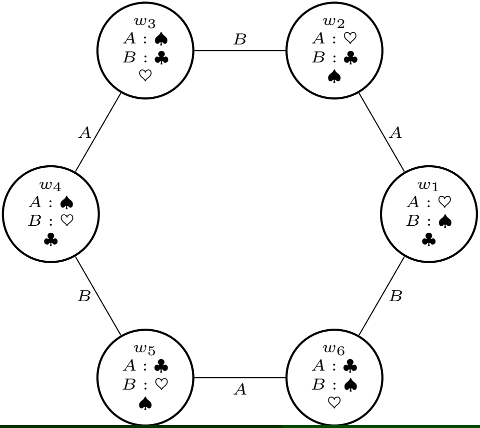
- $\mathcal{K} \models C_G\varphi \to E_G\varphi$
- $\mathcal{K} \models E_G\varphi \to K_a\varphi$ where $a \in G$
- $\mathcal{K} \models K_a\varphi \to D_G\varphi$ where $a \in G$
- $\mathcal{K} \models C_G\varphi \to E_G E_G\varphi$
  $\mathcal{K} \models C_G\varphi \to E_G E_G E_G\varphi$
  $\dots$
- $\mathcal{K} \models C_G\varphi \to E_G C_G\varphi$

Example

We have a deck with tree cards $\heartsuit$, $\clubsuit$ and $\spadesuit$, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.

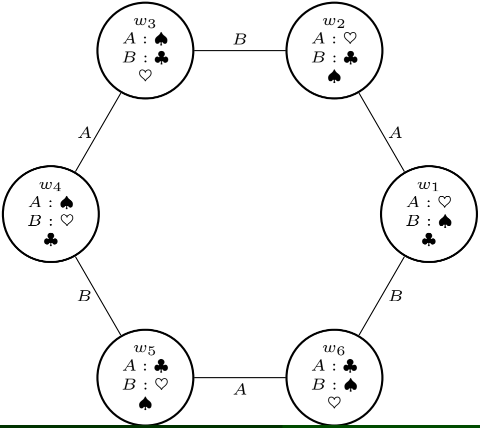
Example

We have a deck with tree cards ♡, ♣ and ♠, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.



Example

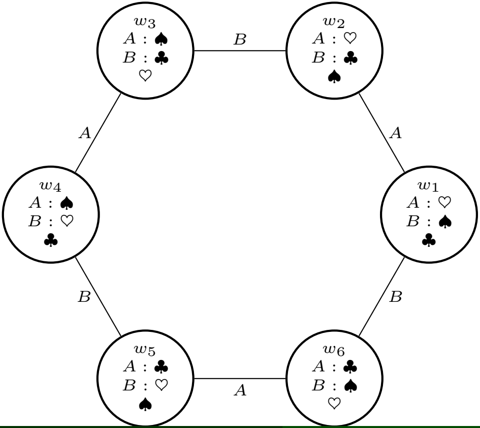
We have a deck with tree cards $\heartsuit$, $\clubsuit$ and $\spadesuit$, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.



$$M, w_1 \vDash K_A(\clubsuit \vee \spadesuit)$$

Example

We have a deck with tree cards $\heartsuit$, $\clubsuit$ and $\spadesuit$, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.

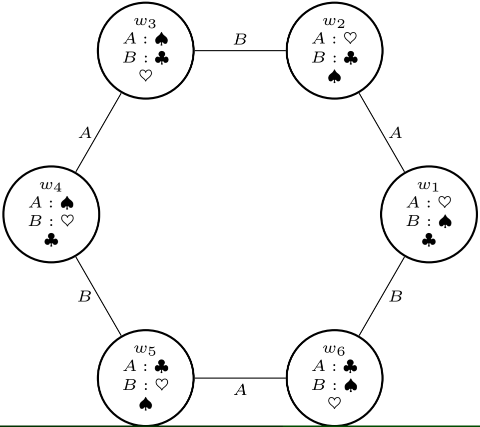


$$M, w_1 \models K_A(\clubsuit \vee \spadesuit)$$

$$M, w_1 \models K_B(\clubsuit \vee \heartsuit)$$

Example

We have a deck with tree cards ♡, ♣ and ♠, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.



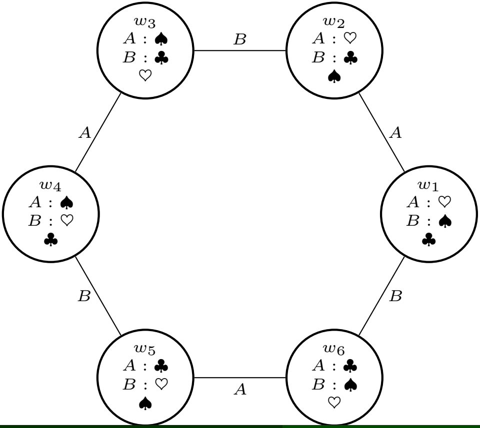
$$M, w_1 \vDash K_A(\clubsuit \vee \spadesuit)$$

$$M, w_1 \vDash K_B(\clubsuit \vee \heartsuit)$$

$$M, w_1 \vDash \neg K_B(\clubsuit)$$

Example

We have a deck with tree cards $\heartsuit$, $\clubsuit$ and $\spadesuit$, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.



$$M, w_1 \models K_A(\clubsuit \vee \spadesuit)$$

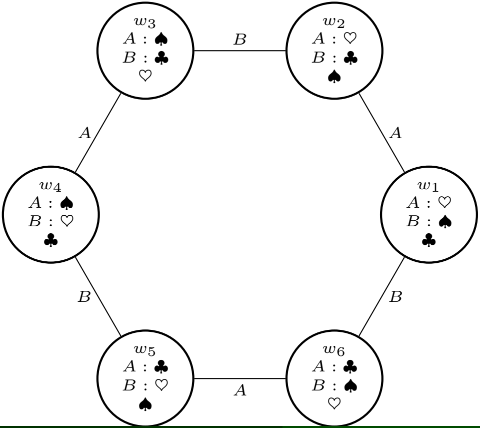
$$M, w_1 \models K_B(\clubsuit \vee \heartsuit)$$

$$M, w_1 \models \neg K_B(\clubsuit)$$

$$M, w_1 \models \neg K_A(\clubsuit)$$

Example

We have a deck with tree cards $\heartsuit$, $\clubsuit$ and $\spadesuit$, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.



$$M, w_1 \models K_A(\clubsuit \vee \spadesuit)$$

$$M, w_1 \models K_B(\clubsuit \vee \heartsuit)$$

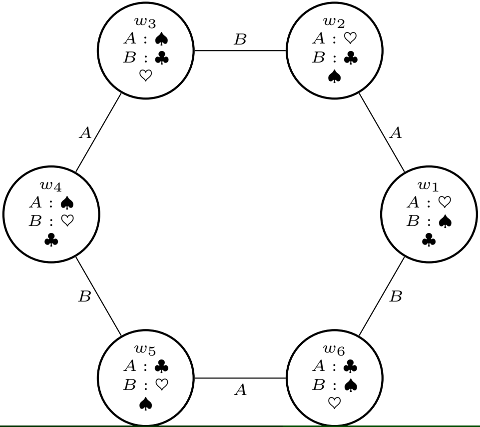
$$M, w_1 \models \neg K_B(\clubsuit)$$

$$M, w_1 \models \neg K_A(\clubsuit)$$

$$M, w_1 \models D_{\{A,B\}}(\clubsuit)$$

Example

We have a deck with tree cards ♡, ♣ and ♠, we take one card and put covered on the table. We give the remaining two cards one by one to the players: Alice and Bob. Each player knows his card and the deck.



$$M, w_1 \models K_A(\clubsuit \vee \spadesuit)$$

$$M, w_1 \models K_B(\clubsuit \vee \heartsuit)$$

$$M, w_1 \models \neg K_B(\clubsuit)$$

$$M, w_1 \models \neg K_A(\clubsuit)$$

$$M, w_1 \models D_{\{A,B\}}(\clubsuit)$$

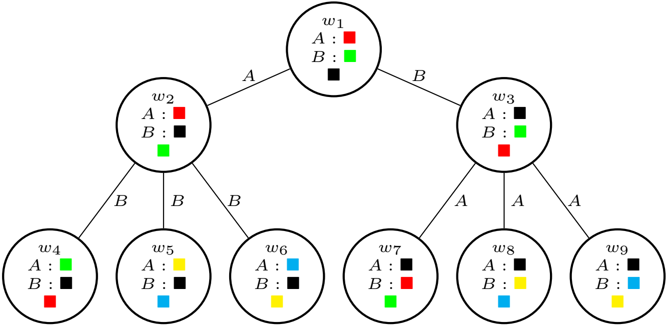
$$M \models C_{\{A,B\}}(\heartsuit \vee \spadesuit \vee \clubsuit)$$

Example

We have two deck with colored cards and two players, Alice and Bob.
The first deck, A, contains ■, ■ and ■.
The second deck, B, contains ■, ■ and ■.
We take a deck randomly, players doesn't know which one, and we give one card for each player. Each player knows his card and the decks.

Example

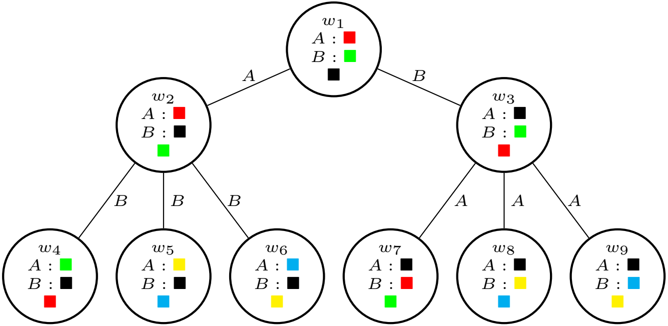
We have two deck with colored cards and two players, Alice and Bob.
The first deck, A, contains 🟥, 🟩 and ⬛.
The second deck, B, contains 🟨, 🟦 and ⬛.
We take a deck randomly, players doesn't know which one, and we give one card for each player. Each player knows his card and the decks.



Example

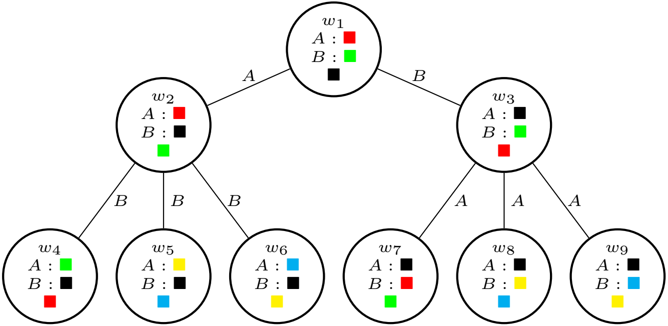
We have two deck with colored cards and two players, Alice and Bob.
The first deck, A, contains $\color{red}\blacksquare$, $\color{lime}\blacksquare$ and $\blacksquare$.
The second deck, B, contains $\color{yellow}\blacksquare$, $\color{cyan}\blacksquare$ and $\blacksquare$.
We take a deck randomly, players doesn't know which one, and we give one card for each player. Each player knows his card and the decks.

$$M, w_1 \models K_A(\neg {\color{cyan}\blacksquare})$$



Example

We have two deck with colored cards and two players, Alice and Bob.
The first deck, A, contains ■, ■ and ■.
The second deck, B, contains ■, ■ and ■.
We take a deck randomly, players doesn't know which one, and we give one card for each player. Each player knows his card and the decks.

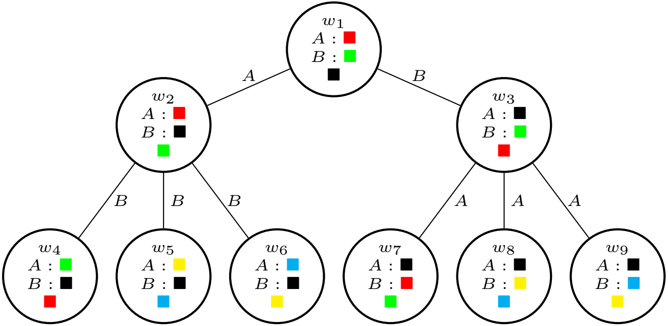


$$M, w_1 \vDash K_A(\neg ■)$$

$$M, w_1 \vDash K_B(\neg ■)$$

Example

We have two deck with colored cards and two players, Alice and Bob.
The first deck, A, contains ■, ■ and ■.
The second deck, B, contains ■, ■ and ■.
We take a deck randomly, players doesn't know which one, and we give one card for each player. Each player knows his card and the decks.



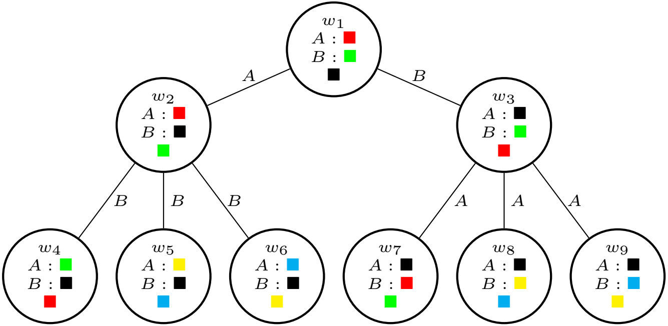
$$M, w_1 \models K_A(\neg \blacksquare)$$

$$M, w_1 \models K_B(\neg \blacksquare)$$

$$M, w_1 \models E_{\{A,B\}}(\neg \blacksquare)$$

Example

We have two deck with colored cards and two players, Alice and Bob.
The first deck, A, contains 🟥, 🟩 and ⬛.
The second deck, B, contains 🟨, 🟦 and ⬛.
We take a deck randomly, players doesn't know which one, and we give one card for each player. Each player knows his card and the decks.



$$M, w_1 \models K_A(\neg {\color{cyan}\blacksquare})$$

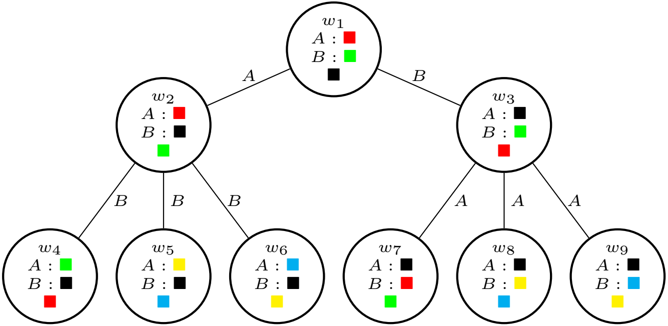
$$M, w_1 \models K_B(\neg {\color{cyan}\blacksquare})$$

$$M, w_1 \models E_{\{A,B\}}(\neg {\color{cyan}\blacksquare})$$

$$M, w_1 \not\models K_A K_B(\neg {\color{cyan}\blacksquare})$$

Example

We have two deck with colored cards and two players, Alice and Bob.
The first deck, A, contains ■, ■ and ■.
The second deck, B, contains ■, ■ and ■.
We take a deck randomly, players doesn't know which one, and we give one card for each player. Each player knows his card and the decks.



$$M, w_1 \models K_A(\neg \blacksquare)$$

$$M, w_1 \models K_B(\neg \blacksquare)$$

$$M, w_1 \models E_{\{A,B\}}(\neg \blacksquare)$$

$$M, w_1 \not\models K_A K_B(\neg \blacksquare)$$

$$M, w_1 \not\models C_{\{A,B\}}(\neg \blacksquare)$$

Dynamic Epistemic Logic

What is missing

Until now we are able to reason about

- Knowledge of agents
- Knowledge of groups of agents

What is missing

Until now we are able to reason about

- Knowledge of agents
- Knowledge of groups of agents

But

- The knowledge is static, only deduction, no investigation or speaking
- If something in the problem change I have to create a new model

What is missing

Until now we are able to reason about

- Knowledge of agents
- Knowledge of groups of agents

But

- The knowledge is static, only deduction, no investigation or speaking
- If something in the problem change I have to create a new model

We need *epistemic actions*!

Example

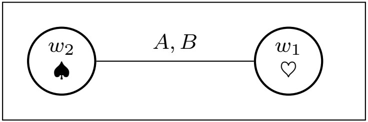
We have a deck with two types of card $\heartsuit$ and $\spadesuit$, we take one card and put it covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

Then Alice reach the table, look the card and let it covered on the table, all in front of Bob.

Example

We have a deck with two types of card ♡ and ♠, we take one card and put it covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

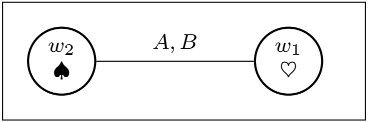
Then Alice reach the table, look the card and let it covered on the table, all in front of Bob.

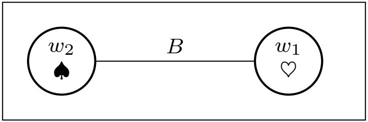


Example

We have a deck with two types of card ♡ and ♠, we take one card and put it covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

Then Alice reach the table, look the card and let it covered on the table, all in front of Bob.





Example

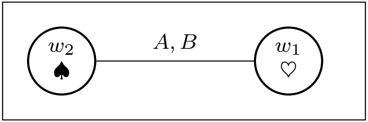
We have a deck with two types of card $\heartsuit$ and $\spadesuit$, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

Then someone reaches the table and reveal the card.

Example

We have a deck with two types of card $\heartsuit$ and $\spadesuit$, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

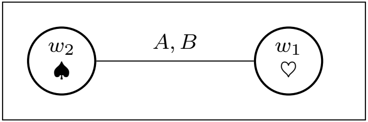
Then someone reaches the table and reveal the card.



Example

We have a deck with two types of card ♡ and ♠, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

Then someone reaches the table and reveal the card.



Example

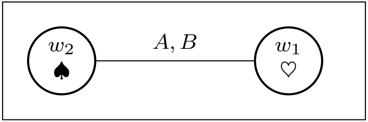
We have a deck with two types of card ♡ and ♠, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

Then Bob leaves the room, Alice can look the card or not.

Example

We have a deck with two types of card ♡ and ♠, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

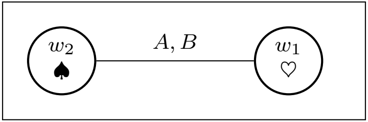
Then Bob leaves the room, Alice can look the card or not.

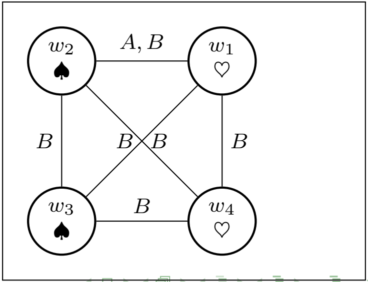


Example

We have a deck with two types of card ♡ and ♠, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules.

Then Bob leaves the room, Alice can look the card or not.





Example

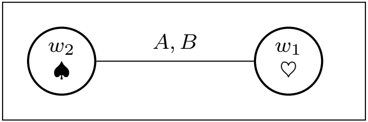
We have a deck with two types of card ♡ and ♠, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules, they are outside the room.

Alice and Bob, one by one, enter the room, each can look the card or not.

Example

We have a deck with two types of card ♡ and ♠, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules, they are outside the room.

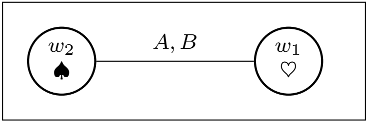
Alice and Bob, one by one, enter the room, each can look the card or not.

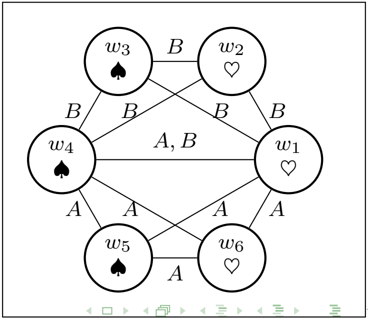


Example

We have a deck with two types of card ♡ and ♠, we take one card and put covered on the table. We have two players, Alice and Bob, they don't know the card but they know the rules, they are outside the room.

Alice and Bob, one by one, enter the room, each can look the card or not.





Epistemic Actions

Epistemic Actions

We need some model for epistemic actions

Epistemic Actions

We need some model for epistemic actions

We use **Kripke model** also as **action model**

Epistemic Actions

We need some model for epistemic actions

We use **Kripke model** also as **action model**

Each action defines a way of updating the model of the system

Epistemic Actions

We need some model for epistemic actions

We use **Kripke model** also as **action model**

Each action defines a way of updating the model of the system

E.g. Public announcement

Action Model

Given At and a logical language $\mathcal{L}$ we define an **action model** U as
$U = (S, R, pre)$
where

Action Model

Given At and a logical language $\mathcal{L}$ we define an **action model** U as
$U = (S, R, pre)$
where

S is a set of action points

Action Model

Given At and a logical language $\mathcal{L}$ we define an **action model** U as
$U = (S, R, pre)$
where

- S is a set of action points
- R : $Ag \to S \times S$ is a function yielding an accessibility relation R_a for each agent a.

Action Model

Given *At* and a logical language $\mathcal{L}$ we define an **action model** U as
$U = (S, R, pre)$
where

- S is a set of action points
- R : $Ag \to S \times S$ is a function yielding an accessibility relation R_a for each agent a.
- pre : $S \to \mathcal{L}$ is a precondition function that assign a precondition $pre(\sigma) \in \mathcal{L}$ to each $\sigma \in S$

An **epistemic action** is a pointed action model (U, σ) with $\sigma \in S$.

Action execution

Let $M = (W, R, V)$ be a Kripke model and $U = (S, R', pre)$ be an action model. We define $M \otimes U$ to be M' such that

Action execution

Let $M = (W, R, V)$ be a Kripke model and $U = (S, R', pre)$ be an action model. We define $M \otimes U$ to be M' such that

- The set of possible worlds is

$$\{(w, \alpha) \in W \times S \mid M, w \vDash pre(\alpha)\}$$

Action execution

Let $M = (W, R, V)$ be a Kripke model and $U = (S, R', pre)$ be an action model. We define $M \otimes U$ to be M' such that

- The set of possible worlds is

$$\{(w, \alpha) \in W \times S \mid M, w \vDash pre(\alpha)\}$$

- The function yielding the accessible relations is

$$a \mapsto \{((w, \alpha), (w', \alpha')) \mid (w, w') \in R_a \wedge (\alpha, \alpha') \in R'_a\}$$

Action execution

Let $M = (W, R, V)$ be a Kripke model and $U = (S, R', pre)$ be an action model. We define $M \otimes U$ to be M' such that

- The set of possible worlds is

$$\{(w, \alpha) \in W \times S \mid M, w \vDash pre(\alpha)\}$$

- The function yielding the accessible relations is

$$a \mapsto \{((w, \alpha)\ , (w', \alpha')) \mid (w, w') \in R_a \wedge (\alpha, \alpha') \in R'_a\}$$

- The valuation function is such that $V(w, \alpha) = V(w)$

Dynamic Epistemic Logic Language

We extend our language $\mathcal{L} = L(At, Op, Ag)$ with epistemic actions from action model $U = (S, R', pre)$.

Dynamic Epistemic Logic Language

We extend our language $\mathcal{L} = L(At, Op, Ag)$ with epistemic actions from action model $U = (S, R', pre)$.

$\varphi \in \mathcal{L}$ defined by *BNF*

$$\varphi \;=\; p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi$$

Dynamic Epistemic Logic Language

We extend our language $\mathcal{L} = L(At, Op, Ag)$ with epistemic actions from action model $U = (S, R', pre)$.

$\varphi \in \mathcal{L}$ defined by *BNF*

$$\varphi \;=\; p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \;\mid [\alpha]\varphi$$

Dynamic Epistemic Logic Language

We extend our language $\mathcal{L} = L(At, Op, Ag)$ with epistemic actions from action model $U = (S, R', pre)$.

$\varphi \in \mathcal{L}$ defined by *BNF*

$$\varphi \;=\; p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \;\mid [\alpha]\varphi$$

where $p \in$ *At*, $\Box \in$ *Op* (probably K_a, C_a, E_a, D_a for $a \in Ag$) and $\alpha \in S$.

Semantics

Ideally the semantic of proposition $[\alpha]\varphi$ is:

Semantics

Ideally the semantic of proposition $[\alpha]\varphi$ is:

- After we apply the action α, the proposition φ holds.

Semantics

Ideally the semantic of proposition $[\alpha]\varphi$ is:

- After we apply the action α, the proposition φ holds.

Formally

$$M, w \vDash [\alpha]\varphi \;\text{ iff }\; M, w \vDash pre(\alpha) \text{ implies } M \otimes U, (w, \alpha) \vDash \varphi$$

Public Announcement: $!\varphi$

Public Announcement: $!\varphi$

- The simplest action.

Public Announcement: $!\varphi$

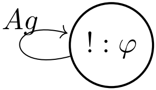
- The simplest action.
- All agents receive the information φ.

Public Announcement: $!\varphi$

- The simplest action.
- All agents receive the information φ.
- All agents know that all other agents did receive the information φ and so on.

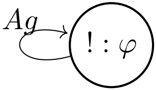
Public Announcement: $!\varphi$

Action model for the announcement of $\varphi \in \mathcal{L}$



Public Announcement: $!\varphi$

Action model for the announcement of $\varphi \in \mathcal{L}$



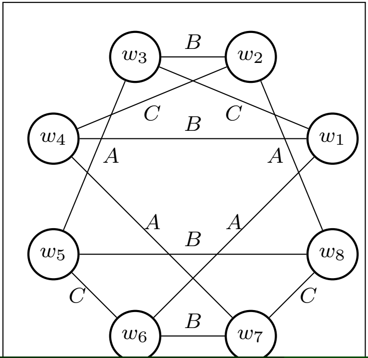
The result of applying this action is simply to remove the possible worlds in which ϕ doesn't hold (**relativization of the model** M **in** M_φ).

Example: muddy children

Three children Alice, Bob and Charlie play in the garden, their mother told them to not get dirty. No one of the children can see if he is dirty on the back or not, but everyone can see all his siblings. Then initially every child knows only the state of the other children.

Example: muddy children

Three children Alice, Bob and Charlie play in the garden, their mother told them to not get dirty. No one of the children can see if he is dirty on the back or not, but everyone can see all his siblings. Then initially every child knows only the state of the other children.



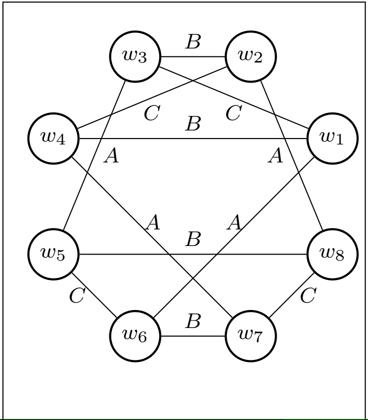
$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_2) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto f\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_5) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_7) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_8) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto f\}$$

Example: muddy children

The mother announces to them that at least one is muddy:
$[!(D_A \vee D_B \vee D_C)]$.

Example: muddy children

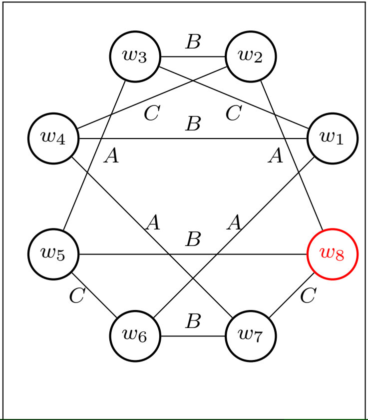
The mother announces to them that at least one is muddy:
$[!(D_A \vee D_B \vee D_C)]$.



$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_2) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto f\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_5) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_7) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_8) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto f\}$$

Example: muddy children

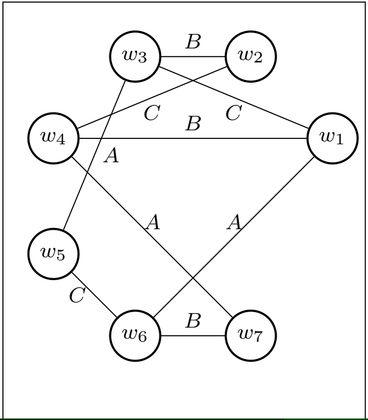
The mother announces to them that at least one is muddy:
$[!(D_A \vee D_B \vee D_C)]$.



$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_2) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto f\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_5) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_7) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_8) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto f\}$$

Example: muddy children

The mother announces to them that at least one is muddy:
$[!(D_A \vee D_B \vee D_C)]$.



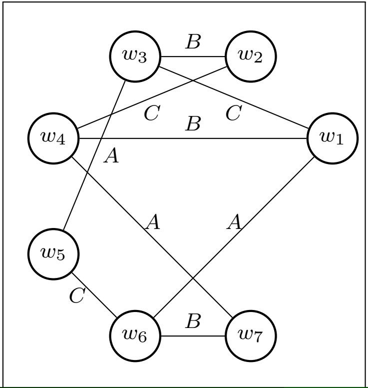
$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_2) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto f\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_5) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_7) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto t\}$$

Example: muddy children

The children all together announce that they still are not able to know if they are dirty or not:
$[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.

Example: muddy children

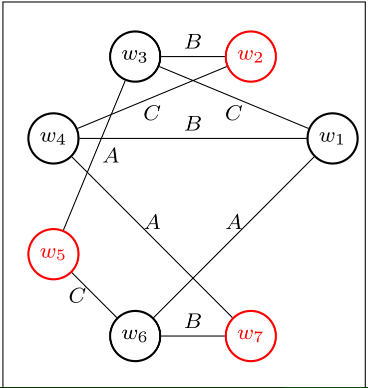
The children all together announce that they still are not able to know if they are dirty or not:
$[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.



$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_2) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto f\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_5) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_7) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto t\}$$

Example: muddy children

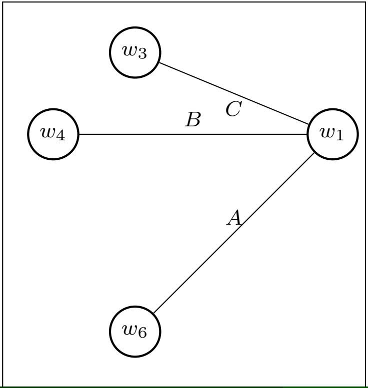
The children all together announce that they still are not able to know if they are dirty or not:
$[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.



$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_2) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto f\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_5) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_7) = \{D_A \mapsto f, D_B \mapsto f, D_C \mapsto t\}$$

Example: muddy children

The children all together announces that they still are no able to know if they are dirty or not:
$[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.



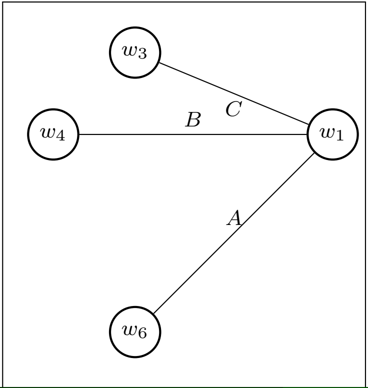
$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$

Example: muddy children

The children take a moment to think about what they know now and announces that they still are no able to know if they are dirty or not: $[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.

Example: muddy children

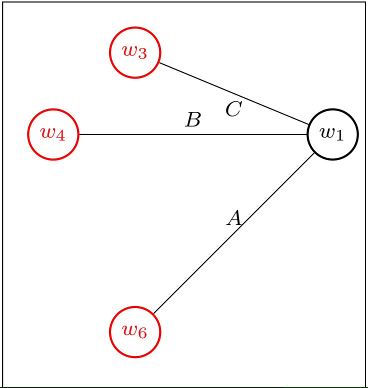
The children take a moment to think about what they know now and announces that they still are no able to know if they are dirty or not: $[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.



$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$
$$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$$
$$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$$
$$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$$

Example: muddy children

The children take a moment to think about what they know now and announces that they still are no able to know if they are dirty or not: $[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.



$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$
$V(w_3) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto f\}$
$V(w_4) = \{D_A \mapsto t, D_B \mapsto f, D_C \mapsto t\}$
$V(w_6) = \{D_A \mapsto f, D_B \mapsto t, D_C \mapsto t\}$

Example: muddy children

The children take a moment to think about what they know now and announces that they still are no able to know if they are dirty or not: $[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.

$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$

Example: muddy children

The children take a moment to think about what they know now and announces that they still are no able to know if they are dirty or not: $[!(\neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C)]$.

w_1

$$V(w_1) = \{D_A \mapsto t, D_B \mapsto t, D_C \mapsto t\}$$

Now each child knows that he and his siblings are all dirty!

Example: muddy children

Then we can say that

$$\varphi_1 = D_A \vee D_B \vee D_C$$
$$\varphi_2 = \neg K_A D_A \wedge \neg K_A \neg D_A \wedge \neg K_B D_B \wedge \neg K_B \neg D_B \wedge \neg K_C D_C \wedge \neg K_C \neg D_C$$

$$M \vDash [!\varphi_1][!\varphi_2][!\varphi_2](K_A D_A \wedge K_B D_B \wedge K_C D_C)$$

Particular sentences

Some sentences are

Particular sentences

Some sentences are

successful: After a public announcement of φ, φ is true, even if the accessibility relation is not an equivalence relation
$\mathcal{K} \vDash [!\varphi]\varphi$

Particular sentences

Some sentences are

successful: After a public announcement of φ, φ is true, even if the accessibility relation is not an equivalence relation
$\mathcal{K} \vDash [!\varphi]\varphi$
E.g. every atomic formula $p \in At$ is successful

Particular sentences

Some sentences are

successful: After a public announcement of φ, φ is true, even if the accessibility relation is not an equivalence relation
$\mathcal{K} \vDash [!\varphi]\varphi$
E.g. every atomic formula $p \in At$ is successful

unsuccessful: After a public announcement of φ, φ is false, even if the accessibility relation is not an equivalence relation
$\mathcal{K} \vDash [!\varphi]\neg\varphi$

Particular sentences

Some sentences are

successful: After a public announcement of φ, φ is true, even if the accessibility relation is not an equivalence relation
$\mathcal{K} \models [!\varphi]\varphi$
E.g. every atomic formula $p \in At$ is successful

unsuccessful: After a public announcement of φ, φ is false, even if the accessibility relation is not an equivalence relation
$\mathcal{K} \models [!\varphi]\neg\varphi$
E.g. (from the Moore's paradox) $(p \wedge \neg K(p))$ is unsuccessful

Public announcement and common knowledge

Public announcement and common knowledge seems to be related somehow

Public announcement and common knowledge

Public announcement and common knowledge seems to be related somehow

If I announce in public φ then it will be common knowledge, isn't it?

Public announcement and common knowledge

Public announcement and common knowledge seems to be related somehow

If I announce in public φ then it will be common knowledge, isn't it?

Well, not exactly!

Public announcement and common knowledge

Public announcement and common knowledge seems to be related somehow

If I announce in public φ then it will be common knowledge, isn't it?

Well, not exactly!
There are problem with the public announcement of statement about knowledge, like in the Moore's paradox

Public announcement and common knowledge

Public announcement and common knowledge seems to be related somehow

If I announce in public φ then it will be common knowledge, isn't it?

Well, not exactly!
There are problem with the public announcement of statement about knowledge, like in the Moore's paradox

But we can state the fact that:
If φ is successful ($\mathcal{K} \vDash [!\varphi]\varphi$) then also $\mathcal{K} \vDash [!\varphi]C\varphi$

Epistemic Logic for Security

Idea

- A lot (not all) security properties specify what each agent must know and mustn't know
  - confidentiality
  - agent authentication
  - data authentication
  - anonymity

Idea

- A lot (not all) security properties specify what each agent must know and mustn't know
  - confidentiality
  - agent authentication
  - data authentication
  - anonymity
- Ideally epistemic logic should be a good tool for speaking about security
  - formal proof systems
  - higher level analysis (better then bisimulation)
  - knowledge and ability
  - knowledge and time
  - knowledge and strategy (epistemic foundation of game theory)

Uses

There are two possible uses of epistemic logic in security

Uses

There are two possible uses of epistemic logic in security

Definitional: Logic is used to formalize properties we want protocol to satisfy

Uses

There are two possible uses of epistemic logic in security

Definitional: Logic is used to formalize properties we want protocol to satisfy

Several successes!

Uses

There are two possible uses of epistemic logic in security

Definitional: Logic is used to formalize properties we want protocol to satisfy

Several successes!

Practical: Logic is used to verify properties of a protocol or to derive an attack

Uses

There are two possible uses of epistemic logic in security

Definitional: Logic is used to formalize properties we want protocol to satisfy

Several successes!

Practical: Logic is used to verify properties of a protocol or to derive an attack

Still fewer successes...

Problems

Problems

- Formalization of protocols as sequence of epistemic actions is not simple

Problems

- Formalization of protocols as sequence of epistemic actions is not simple
- Verification of epistemic properties tends to be expensive

Problems

- Formalization of protocols as sequence of epistemic actions is not simple
- Verification of epistemic properties tends to be expensive
- We want some simplification for cryptography

Problems

- Formalization of protocols as sequence of epistemic actions is not simple
- Verification of epistemic properties tends to be expensive
- We want some simplification for cryptography
- Usually we want to use the formalization of the protocol in some process algebra

Problems

- Formalization of protocols as sequence of epistemic actions is not simple
- Verification of epistemic properties tends to be expensive
- We want some simplification for cryptography
- Usually we want to use the formalization of the protocol in some process algebra
- Sometimes we use multimodal logic, like temporal-epistemic logic

Problems

- Formalization of protocols as sequence of epistemic actions is not simple
- Verification of epistemic properties tends to be expensive
- We want some simplification for cryptography
- Usually we want to use the formalization of the protocol in some process algebra
- Sometimes we use multimodal logic, like temporal-epistemic logic

We already know BAN logic

Spatial-Epistemic Logic

Idea

Idea

- process calculus models $+$ logic specification (instead of bisimulation)

Idea

- process calculus models + logic specification (instead of bisimulation)
- dynamic spatial logic substitute an explicit definition of the agents (where is the knowledge instead of whose)

Idea

- process calculus models + logic specification (instead of bisimulation)
- dynamic spatial logic substitute an explicit definition of the agents (where is the knowledge instead of whose)
- temporal fragment for speaking about protocols

Idea

- process calculus models + logic specification (instead of bisimulation)
- dynamic spatial logic substitute an explicit definition of the agents (where is the knowledge instead of whose)
- temporal fragment for speaking about protocols
- process calculus based on π-calculus, but with lots of freedom on messages (customization)

Idea

- process calculus models + logic specification (instead of bisimulation)
- dynamic spatial logic substitute an explicit definition of the agents (where is the knowledge instead of whose)
- temporal fragment for speaking about protocols
- process calculus based on π-calculus, but with lots of freedom on messages (customization)
- automatic derivation of *Dolev-Yao* attacker

Model

Model

- We start from π-calculus

Model

- We start from π-calculus
- We extend with the capacity of communicate arbitrary structured terms, defined by a term algebra (like applied π-calculus)

Model

- We start from π-calculus
- We extend with the capacity of communicate arbitrary structured terms, defined by a term algebra (like applied π-calculus)
- To specify the model we need a Signature Σ and a set of equations E

Terms and Equational Theories

We define

Terms and Equational Theories

We define

- A set of variables $x, y, z \in Var$

Terms and Equational Theories

We define

- A set of variables $x, y, z \in Var$
- A set of names $m, n \in \Lambda$

Terms and Equational Theories

We define

- A set of variables $x, y, z \in Var$
- A set of names $m, n \in \Lambda$
- A signature Σ with $f/n \in \Sigma$ pair function symbol with arity

Terms and Equational Theories

We define

- A set of variables $x, y, z \in Var$
- A set of names $m, n \in \Lambda$
- A signature Σ with $f/n \in \Sigma$ pair function symbol with arity
- The set of terms $s, t, v \in Terms =$
  $\Lambda \cup Var \cup \bigcup_{f/n \in \Sigma} \{f(t_1, \ldots, t_n) \mid \{t_1, \ldots, t_n\} \subseteq Terms\}$

Terms and Equational Theories

We define

- A set of variables $x, y, z \in Var$
- A set of names $m, n \in \Lambda$
- A signature Σ with $f/n \in \Sigma$ pair function symbol with arity
- The set of terms $s, t, v \in Terms = \Lambda \cup Var \cup \bigcup_{f/n \in \Sigma} \{f(t_1, \ldots, t_n) \mid \{t_1, \ldots, t_n\} \subseteq Terms\}$
- An equational theory E to define the semantics of function symbols in Σ

Equational Theory as Set of Rewrite Rule

Equational Theory as Set of Rewrite Rule

- E is a set of equation between terms $t = s$

Equational Theory as Set of Rewrite Rule

- E is a set of equation between terms $t = s$
- We choose to see E as a set of rewrite rules by orienting each rule $t \to s$

Equational Theory as Set of Rewrite Rule

- E is a set of equation between terms $t = s$
- We choose to see E as a set of rewrite rules by orienting each rule $t \to s$
- We say that $t =_E s$ iff exists v such that $t \to^* v$ and $s \to^* v$

Equational Theory as Set of Rewrite Rule

- E is a set of equation between terms $t = s$
- We choose to see E as a set of rewrite rules by orienting each rule $t \to s$
- We say that $t =_E s$ iff exists v such that $t \to^* v$ and $s \to^* v$
- E is subterm convergent if $t = s$ only if s is a proper subterm of t

Equational Theory as Set of Rewrite Rule

- E is a set of equation between terms $t = s$
- We choose to see E as a set of rewrite rules by orienting each rule $t \rightarrow s$
- We say that $t =_E s$ iff exists v such that $t \rightarrow^* v$ and $s \rightarrow^* v$
- E is subterm convergent if $t = s$ only if s is a proper subterm of t
- If E is subterm convergent then $t \rightarrow^* s$ iff $t =^* s$

Equational Theory as Set of Rewrite Rule

- E is a set of equation between terms $t = s$
- We choose to see E as a set of rewrite rules by orienting each rule $t \rightarrow s$
- We say that $t =_E s$ iff exists v such that $t \rightarrow^* v$ and $s \rightarrow^* v$
- E is subterm convergent if $t = s$ only if s is a proper subterm of t
- If E is subterm convergent then $t \rightarrow^* s$ iff $t =^* s$
- We will assume that E is subterm convergent

Constructors and Destructors

Given Σ and E we define

Constructors and Destructors

Given Σ and E we define

Destructor: Each function symbol that is the outermost function symbol in the left part of a rewrite rule in E

Constructors and Destructors

Given Σ and E we define

Destructor: Each function symbol that is the outermost function symbol in the left part of a rewrite rule in E

Constructor: Any other function symbol in Σ

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) = x\}$

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) = x\}$

We have

- $E = \{dec(enc(x, y), y) \to x\}$
- destructors$(E) = \{dec\}$
- constructors$(E) = \{enc\}$

Dolev-Yao equational closure

Represents all possible information that may be produced by a set of therms while following the rules of the equational theory

Dolev-Yao equational closure

Represents all possible information that may be produced by a set of therms while following the rules of the equational theory

The Dolev-Yao equational closure of a set of therms ψ is the least set of therms $\mathcal{F}(\psi)$ such that

- $\psi \subseteq \mathcal{F}(\psi)$
- $\forall f/n \in \Sigma$. if $f \in$ constructor(E) $\wedge t_1, \ldots, t_n \in \mathcal{F}(\psi)$ then $f(t_1, \ldots, t_n) \in \mathcal{F}(\psi)$
- $\forall f/n \in \Sigma$. if $f \in$ destructor(E) $\wedge t_1, \ldots, t_n \in \mathcal{F}(\psi) \wedge f(t_1, \ldots, t_n) \to t'$ then $\in \mathcal{F}(\psi)$

Dolev-Yao equational closure

Represents all possible information that may be produced by a set of therms while following the rules of the equational theory

The Dolev-Yao equational closure of a set of therms ψ is the least set of therms $\mathcal{F}(\psi)$ such that

- $\psi \subseteq \mathcal{F}(\psi)$
- $\forall f/n \in \Sigma$. if $f \in$ constructor(E) $\wedge t_1, \ldots, t_n \in \mathcal{F}(\psi)$ then $f(t_1, \ldots, t_n) \in \mathcal{F}(\psi)$
- $\forall f/n \in \Sigma$. if $f \in$ destructor(E) $\wedge t_1, \ldots, t_n \in \mathcal{F}(\psi) \wedge f(t_1, \ldots, t_n) \rightarrow t'$ then $\in \mathcal{F}(\psi)$

If $\varphi \in \mathcal{F}(\psi)$ then we write $\psi \Vdash \varphi$ (Knowledge derivation)

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) \to x\}$
- $\Lambda = k_1, k_2, m$

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) \to x\}$
- $\Lambda = k_1, k_2, m$

We have that $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \Vdash m$

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) \to x\}$
- $\Lambda = k_1, k_2, m$

We have that $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \Vdash m$

Proof (where $\psi = \{k_2, enc(k_1, k_2), enc(m, k_1)\}$)

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) \to x\}$
- $\Lambda = k_1, k_2, m$

We have that $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \Vdash m$

Proof (where $\psi = \{k_2, enc(k_1, k_2), enc(m, k_1)\}$)

- $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \subseteq \mathcal{F}(\psi)$

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) \to x\}$
- $\Lambda = k_1, k_2, m$

We have that $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \Vdash m$

Proof (where $\psi = \{k_2, enc(k_1, k_2), enc(m, k_1)\}$)

- $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \subseteq \mathcal{F}(\psi)$
- $\{k_1\} \in \mathcal{F}(\psi)$ since $dec \in$ destructor(E) and $enc(k_1, k_2) \in \mathcal{F}(\psi)$ and $k_2 \in \mathcal{F}(\psi)$ and $dec(enc(k_1, k_2), k_2) \to k_1 \in E$

Example

Given

- $\Sigma = \{enc/2, dec/2\}$
- $E = \{dec(enc(x, y), y) \to x\}$
- $\Lambda = k_1, k_2, m$

We have that $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \Vdash m$

Proof (where $\psi = \{k_2, enc(k_1, k_2), enc(m, k_1)\}$)

- $\{k_2, enc(k_1, k_2), enc(m, k_1)\} \subseteq \mathcal{F}(\psi)$
- $\{k_1\} \in \mathcal{F}(\psi)$ since $dec \in$ destructor(E) and $enc(k_1, k_2) \in \mathcal{F}(\psi)$ and $k_2 \in \mathcal{F}(\psi)$ and $dec(enc(k_1, k_2), k_2) \to k_1 \in E$
- $\{m\} \in \mathcal{F}(\psi)$ since $dec \in$ destructor(E) and $enc(m, k_1) \in \mathcal{F}(\psi)$ and $k_1 \in \mathcal{F}(\psi)$ and $dec(enc(m, k_1), k_1) \to m \in E$

Process Calculus

Processes P, Q are defined in BNF

$$
\begin{aligned}
P, Q ::= {} & 0 && \textit{(Null Process)} \\
\mid {} & P|Q && \textit{(Parallel Composition)} \\
\mid {} & (\nu n)P && \textit{(Name Restriction)} \\
\mid {} & \alpha.P && \textit{(Action Prefix)} \\
\mid {} & P + Q && \textit{(Choice)} \\
\mid {} & \texttt{let } x = T \texttt{ in } P && \textit{(Let Construct)}
\end{aligned}
$$

Process Calculus

$$
\begin{aligned}
\alpha ::= {} & m(x) && \textit{(Input)} \\
\mid {} & m\langle T \rangle && \textit{(Output)} \\
\mid {} & m\langle * \rangle && \textit{(Attacker Output)} \\
\mid {} & [T_1 = T_2] && \textit{(Test)}
\end{aligned}
$$

$$
\begin{aligned}
T ::= {} & n && \textit{(Name)} \\
\mid {} & x && \textit{(Variable)} \\
\mid {} & f(T_1, \ldots, T_n) && \textit{(Function)}
\end{aligned}
$$

Structural Congruence

if $n \notin fn(P) \cup fv(P)$ then $P|(\nu n)Q \equiv (\nu n)(P|Q)$

$(\nu n)0 \equiv 0$

$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$

if $M_1 =_E M_1'$ then `let x =` M_1 `in` $P \equiv$ `let x =` M_1' `in` P

if $M_1 =_E M_1'$ then $m\langle M_1 \rangle.P \equiv m\langle M_1' \rangle.P$

if $M_1 =_E M_1'$ then $[M_1 = M].P \equiv [M_1' = M].P$

$P|0 \equiv P$

$P|Q \equiv Q|P$

$P|(Q|R) \equiv (P|Q)|R$

$P + Q \equiv Q + P$

$P + (Q + R) \equiv (P + Q) + R$

$[M_1 = M_2].P \equiv [M_2 = M_1].P$

Structural Congruence

if $n \notin fn(P) \cup fv(P)$ then $P|(\nu n)Q \equiv (\nu n)(P|Q)$

$(\nu n)0 \equiv 0$

$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$

if $M_1 =_E M_1'$ then `let x =` M_1 `in` $P \equiv$ `let x =` M_1' `in` P

if $M_1 =_E M_1'$ then $m\langle M_1\rangle.P \equiv m\langle M_1'\rangle.P$

if $M_1 =_E M_1'$ then $[M_1 = M].P \equiv [M_1' = M].P$

$P|0 \equiv P$

$P|Q \equiv Q|P$

$P|(Q|R) \equiv (P|Q)|R$

$P + Q \equiv Q + P$

$P + (Q + R) \equiv (P + Q) + R$

$[M_1 = M_2].P \equiv [M_2 = M_1].P$

Reduction Semantics

Computational steps **inside** a process (no interaction with the environment)

$$\textit{(Let)}\ \frac{M \text{ is destructor free}}{\texttt{let x = } M \texttt{ in } P \longrightarrow P\{x \leftarrow M\}}$$

$$\textit{(Sync)}\ \frac{M \text{ is destructor free}}{n\langle M\rangle.P + R \mid n(x).Q + S \longrightarrow P \mid Q\{x \leftarrow M\}}$$

$$\textit{(Test)}\ \frac{M_1 \text{ and } M_2 \text{ is destructor free} \qquad M_1 =_E M_2}{[M_1 = M_2].P \longrightarrow P}$$

Reduction Semantics

$$\textit{(Par)}\ \frac{P \to Q}{P|R \to Q|R} \qquad \textit{(Scope)}\ \frac{P \to Q}{(\nu n)P \to (\nu n)Q}$$

$$\textit{(Cong)}\ \frac{P \equiv P' \qquad P' \to Q' \qquad Q \equiv Q'}{P \to Q}$$

$$\textit{(Attacker)}\ \frac{M \in \mathcal{F}(gt(Q) \cup \bar{n}) \text{ with } \bar{n} \text{ fresh names}}{c(x).P + R \mid c\langle * \rangle.Q + S \longrightarrow (\nu \bar{n})(P\{x \leftarrow M\}|Q)}$$

Labelled Transition Semantics

Communication with the **environment**

$$\textit{(Tau)}\ \frac{P \to Q}{P \xrightarrow{\tau} Q}$$

$$\textit{(Out)}\ \frac{M \text{ is destructor free}}{n\langle M\rangle.P \xrightarrow{n\langle M\rangle} P}$$

$$\textit{(Inp)}\ \frac{M \text{ is destructor free}}{n(x).P \xrightarrow{n(M)} P}$$

$$\textit{(Attacker Out)}\ \frac{M \in \mathcal{F}(gt(Q) \cup \bar{n}) \text{ with } \bar{n} \text{ fresh names}}{c\langle *\rangle.P \xrightarrow{\nu\bar{n}.c\langle M\rangle} P}$$

Labelled Transition Semantics

$$\textit{(Res)}\ \frac{P \xrightarrow{\alpha} Q \qquad \forall n \in \bar{u}.n \notin \mathsf{names}\alpha}{(\nu\bar{u})P \xrightarrow{\alpha} (\nu\bar{u})Q}$$

$$\textit{(Bound Out)}\ \frac{P \xrightarrow{n\langle M\rangle} P' \qquad \bar{s} \subseteq \mathsf{names}(M) \text{ and } \bar{s} \subseteq \bar{u} \qquad \bar{u'} = \bar{u} \setminus \bar{s}}{(\nu\bar{u})P \xrightarrow{\nu\bar{s}.n\langle M\rangle} (\nu\bar{u'})P'}$$

$$\textit{(Cong)}\ \frac{P \equiv P' \qquad P' \xrightarrow{\alpha} Q' \qquad Q \equiv Q'}{P \xrightarrow{\alpha} Q}$$

Logic

Processes P, Q are defined in BNF

$$
\begin{aligned}
A, B ::= {} & T && \textit{(True)} \\
\mid {} & \neg A && \textit{(Negation)} \\
\mid {} & A \wedge B && \textit{(Conjunction)} \\
\mid {} & 0 && \textit{(Void)} \\
\mid {} & A|B && \textit{(Composition)} \\
\mid {} & H_x.A && \textit{(Hidden Quantification)} \\
\mid {} & \alpha.A && \textit{(Action)} \\
\mid {} & \Box A && \textit{(Always)} \\
\mid {} & \Diamond A && \textit{(Eventually)} \\
\mid {} & @n && \textit{(free name predicate)} \\
\mid {} & K\varphi && \textit{(Knowledge)} \\
\mid {} & Sx.A && \textit{(Secret Quantification)}
\end{aligned}
$$

where

$$
\begin{aligned}
\varphi, \psi ::= {} & \varphi \wedge \psi && \textit{(Conjunction)} \\
\mid {} & t && \textit{(Term)} \\
\mid {} & \top && \textit{(True)}
\end{aligned}
$$

Semantics

$$
\begin{aligned}
&P \vDash T && \text{iff True}\\
&P \vDash \neg A && \text{iff not } \vDash A\\
&P \vDash A \wedge B && \text{iff } P \vDash A \text{ and } P \vDash B\\
&P \vDash 0 && \text{iff } P \equiv 0\\
&P \vDash A|B && \text{iff } \exists Q, R.\ P \equiv Q|R \text{ and } Q \vDash A \text{ and } R \vDash B\\
&P \vDash H_x.A && \text{iff } \exists Q.\ P \equiv (\nu n)Q \text{ and } Q \vDash A\{x \leftarrow n\}\\
&P \vDash \alpha.A && \text{iff } \exists Q.\ P \xrightarrow{\alpha} Q \text{ and } Q \vDash A
\end{aligned}
$$

Semantics

$$
\begin{aligned}
P &\vDash \Box A && \text{iff } \forall Q.\ P \xrightarrow{\tau}^{*} Q \text{ then } Q \vDash A \\
P &\vDash \Diamond A && \text{iff } \exists Q.\ P \xrightarrow{\tau}^{*} Q \text{ and } Q \vDash A \\
P &\vDash @n && \text{iff } n \in fn(P) \\
P &\vDash K\varphi && \text{iff } P \vdash_k \psi \text{ and } \psi \Vdash \varphi \\
P &\vDash Sx.A && \text{iff } \exists Q, t\ .P \equiv (\nu k)Q \text{ and } Q \vDash A\{x \leftarrow t\} \\
& && \text{and } Q \vdash_k \psi \text{ such that } t \in \psi \text{ and } k \in names(t)
\end{aligned}
$$

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

- We use $P \vdash_k \psi$ to model that P has access to the set of terms ψ

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

- We use $P \vdash_k \psi$ to model that P has access to the set of terms ψ
- We use an accessory function $sub()$ for relevant subterms of a term

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

- We use $P \vdash_k \psi$ to model that P has access to the set of terms ψ
- We use an accessory function $sub()$ for relevant subterms of a term
- We don't want every subterm of messages to be in ψ (relevance)

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

- We use $P \vdash_k \psi$ to model that P has access to the set of terms ψ
- We use an accessory function $sub()$ for relevant subterms of a term
- We don't want every subterm of messages to be in ψ (relevance)
  - Decompose (here the idea is that a destructor cannot hide information),

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

- We use $P \vdash_k \psi$ to model that P has access to the set of terms ψ
- We use an accessory function $sub()$ for relevant subterms of a term
- We don't want every subterm of messages to be in ψ (relevance)
    - Decompose (here the idea is that a destructor cannot hide information),
    - We consider values only terms without destructors

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

- We use $P \vdash_k \psi$ to model that P has access to the set of terms ψ
- We use an accessory function $sub()$ for relevant subterms of a term
- We don't want every subterm of messages to be in ψ (relevance)
  - Decompose (here the idea is that a destructor cannot hide information),
  - We consider values only terms without destructors
  - Avoid to take terms that are not closed,

Subterms

To model the knowledge of a process we need to know the set of terms it can see.

- We use $P \vdash_k \psi$ to model that P has access to the set of terms ψ
- We use an accessory function $sub()$ for relevant subterms of a term
- We don't want every subterm of messages to be in ψ (relevance)
  - Decompose (here the idea is that a destructor cannot hide information),
  - We consider values only terms without destructors
  - Avoid to take terms that are not closed,
- We have to deal with restricted names (we use $\uparrow$)

Relevant Terms for a Process

$$\frac{}{0 \vdash_k \emptyset}$$

$$\frac{P \vdash_k \varphi \qquad Q \vdash_k \psi}{P + Q \vdash_k \varphi \cup \psi}$$

$$\frac{P \vdash_k \varphi \qquad Q \vdash_k \psi}{P|Q \vdash_k \varphi \cup \psi}$$

$$\frac{P \vdash_k \varphi}{n(x).P \vdash_k \varphi}$$

$$\frac{P \vdash_k \varphi}{x\langle M\rangle.P \vdash_k \varphi \cup sub(M)}$$

Relevant Terms for a Process

$$\frac{P\{x \leftarrow M\} \vdash_k \varphi}{\texttt{let } n = M \texttt{ in } P \vdash_k \varphi \cup sub(M)}$$

$$\frac{P \vdash_k \varphi}{(\nu n)P \vdash_k \varphi \uparrow n}$$

$$\frac{P \vdash_k \varphi}{[M = N].P \vdash_k \varphi \cup sub(M) \cup sub(N)}$$

Where: $\psi \uparrow x = \{t \mid t \in \psi \wedge x \notin names(t)\}$

Relevant Subterms of a Term

$$\frac{n \text{ is a name}}{sub(n) = \{n\}} \qquad \frac{x \text{ is a variable}}{sub(x) = \emptyset}$$

$$\frac{f \text{ is a destructor}}{sub(f(t_1, \ldots, t_n)) = sub(t_1) \cup \cdots \cup sub(t_n)}$$

$$\frac{f \text{ is a constructor} \qquad \nexists t_i \text{ with a variable or a destructor}}{sub(f(t_1, \ldots, t_n)) = \{f(t_1, \ldots, t_n)\}}$$

$$\frac{f \text{ is a constructor} \qquad \exists t_i \text{ with a variable or a destructor}}{sub(f(t_1, \ldots, t_n)) = sub(t_1) \cup \cdots \cup sub(t_n)}$$

Example: Knowledge

$\Sigma = \{enc/2, dec/2\}$ destructors$(E) = \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$ constructors$(E) = \{enc\}$
$\Lambda = \{c, k, m\}$

Example: Knowledge

$\Sigma = \{enc/2, dec/2\}$	destructors$(E) = \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$	constructors$(E) = \{enc\}$
$\Lambda = \{c, k, m\}$	

$$c\langle dec(k)\rangle.0 \mid c\langle enc(m, k)\rangle.0 \vDash Km$$

Example: Knowledge

$\Sigma = \{enc/2, dec/2\}$ destructors(E) $= \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$ constructors(E) $= \{enc\}$
$\Lambda = \{c, k, m\}$

$c\langle dec(k)\rangle.0 \mid c\langle enc(m, k)\rangle.0 \vDash Km$

because $c\langle dec(k)\rangle.0 \mid c\langle enc(m, k)\rangle.0 \vdash_k \{k, enc(m, k)\}$

Example: Knowledge

$\Sigma = \{enc/2, dec/2\}$	destructors(E) $= \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$	constructors(E) $= \{enc\}$
$\Lambda = \{c, k, m\}$	

$c\langle dec(k)\rangle.0 \mid c\langle enc(m,k)\rangle.0 \vDash Km$

because $c\langle dec(k)\rangle.0 \mid c\langle enc(m,k)\rangle.0 \vdash_k \{k, enc(m,k)\}$
since $c\langle dec(k)\rangle.0 \ \vdash_k \{k\} \cup \emptyset$
and $c\langle enc(m,k)\rangle.0 \ \vdash_k \{enc(m,k)\} \cup \emptyset$

Example: Knowledge

$\Sigma = \{enc/2, dec/2\}$ destructors$(E) = \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$ constructors$(E) = \{enc\}$
$\Lambda = \{c, k, m\}$

$c\langle dec(k)\rangle.0 \mid c\langle enc(m, k)\rangle.0 \vDash Km$

because $c\langle dec(k)\rangle.0 \mid c\langle enc(m, k)\rangle.0 \vdash_k \{k, enc(m, k)\}$
since $c\langle dec(k)\rangle.0 \ \vdash_k \{k\} \cup \emptyset$
and $c\langle enc(m, k)\rangle.0 \ \vdash_k \{enc(m, k)\} \cup \emptyset$
and $\{k, enc(m, k)\} \Vdash m$

Example: Space

Number of parallel threads

$0 := 0$
$1 := \neg 0 \wedge \neg(\neg 0 \mid \neg 0)$
$2 := \neg 0 \wedge \neg 1 \wedge \neg(\neg 0 \mid \neg 0 \mid \neg 0)$

Example: Space

Number of parallel threads

$$0 := 0$$
$$1 := \neg 0 \wedge \neg(\neg 0 \mid \neg 0)$$
$$2 := \neg 0 \wedge \neg 1 \wedge \neg(\neg 0 \mid \neg 0 \mid \neg 0)$$

$$c\langle dec(k)\rangle.0 \mid c\langle enc(m,k)\rangle.0 \vDash 2$$

Example: Space

Number of parallel threads

$0 := 0$
$1 := \neg 0 \wedge \neg(\neg 0 \mid \neg 0)$
$2 := \neg 0 \wedge \neg 1 \wedge \neg(\neg 0 \mid \neg 0 \mid \neg 0)$

$$c\langle dec(k)\rangle.0 \mid c\langle enc(m,k)\rangle.0 \vDash 2$$

$$c\langle dec(k)\rangle.0 \mid c\langle enc(m,k)\rangle.0 \nvDash 1$$

Example: Space

Number of parallel threads

$0 := 0$
$1 := \neg 0 \wedge \neg(\neg 0 \mid \neg 0)$
$2 := \neg 0 \wedge \neg 1 \wedge \neg(\neg 0 \mid \neg 0 \mid \neg 0)$

$c\langle dec(k)\rangle.0 \mid c\langle enc(m,k)\rangle.0 \vDash 2$

$c\langle dec(k)\rangle.0 \mid c\langle enc(m,k)\rangle.0 \nvDash 1$

$c\langle dec(k)\rangle.0 \vDash 1$

Example: Spatial + Dynamic

The process has a restricted name and it is always the case that it send it through channel m
$P := Hx.\Box m\langle x\rangle.T$

Example: Spatial + Dynamic

The process has a restricted name and it is always the case that it send it through channel m
$P := Hx.\Box m\langle x\rangle.T$

$(\nu n).m\langle n\rangle.0 \vDash P$

Example: Spatial + Dynamic

The process has a restricted name and it is always the case that it send it through channel m
$P := \mathsf{H}x.\Box m\langle x\rangle.T$

$(\nu n).m\langle n\rangle.0 \vDash P$

$(\nu c)(\nu n).c\langle m\rangle.m\langle n\rangle.0 \vDash P$

Example: Spatial + Dynamic

The process has a restricted name and it is always the case that it send it through channel m
$P := Hx.\Box m\langle x\rangle.T$

$(\nu n).m\langle n\rangle.0 \vDash P$

$(\nu c)(\nu n).c\langle m\rangle.m\langle n\rangle.0 \vDash P$

$(\nu c)(\nu n).(m\langle n\rangle.0 + c\langle m\rangle.m\langle n\rangle.0) \vDash P$

Example: Spatial + Dynamic

The process has a restricted name and it is always the case that it send it through channel m
$P := \mathsf{H}x.\Box m\langle x\rangle.T$

$(\nu n).m\langle n\rangle.0 \vDash P$

$(\nu c)(\nu n).c\langle m\rangle.m\langle n\rangle.0 \vDash P$

$(\nu c)(\nu n).(m\langle n\rangle.0 + c\langle m\rangle.m\langle n\rangle.0) \vDash P$

$(\nu c)(\nu n).(m(x).0 + c\langle m\rangle.m\langle n\rangle.0) \nvDash P$

Example: Spatial + Dynamic

The process has a restricted name and it is always the case that it send it through channel m
$P := Hx.\Box m\langle x\rangle.T$

$(\nu n).m\langle n\rangle.0 \vDash P$

$(\nu c)(\nu n).c\langle m\rangle.m\langle n\rangle.0 \vDash P$

$(\nu c)(\nu n).(m\langle n\rangle.0 + c\langle m\rangle.m\langle n\rangle.0) \vDash P$

$(\nu c)(\nu n).(m(x).0 + c\langle m\rangle.m\langle n\rangle.0) \nvDash P$

$(\nu n).m\langle n\rangle.0 \mid (\nu c)(\nu n).c\langle m\rangle.m\langle n\rangle.0 \vDash (P \mid P)$

Example

$\Sigma = \{enc/2, dec/2\}$
$E = \{dec(enc(x, y), y) \to x\}$
$\Lambda = \{tag, c, k, m\}$

destructors(E) $= \{dec\}$
constructors(E) $= \{enc\}$

Example

$\Sigma = \{enc/2, dec/2\}$
$E = \{dec(enc(x, y), y) \to x\}$
$\Lambda = \{tag, c, k, m\}$

destructors(E) $= \{dec\}$
constructors(E) $= \{enc\}$

It is always the case that a thread with free name tag knows the key.
$P := \Box Hkey.(@tag \wedge Kkey \mid T)$

Example

$\Sigma = \{enc/2, dec/2\}$ destructors(E) $= \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$ constructors(E) $= \{enc\}$
$\Lambda = \{tag, c, k, m\}$

It is always the case that a thread with free name tag knows the key.
$P := \Box H key.(@tag \wedge K key \mid T)$

$(\nu k).tag\langle k\rangle.0 \vDash P$

Example

$\Sigma = \{enc/2, dec/2\}$ destructors$(E) = \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$ constructors$(E) = \{enc\}$
$\Lambda = \{tag, c, k, m\}$

It is always the case that a thread with free name tag knows the key.
$P := \Box H key.(@tag \wedge K key \mid T)$

$(\nu k).tag\langle k\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \vDash P$

Example

$\Sigma = \{enc/2, dec/2\}$ destructors(E) $= \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$ constructors(E) $= \{enc\}$
$\Lambda = \{tag, c, k, m\}$

It is always the case that a thread with free name tag knows the key.
$P := \Box Hkey.(@tag \wedge Kkey \mid T)$

$(\nu k).tag\langle k\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \mid (\nu k).tag\langle k\rangle.0 \vDash P$

Example

$\Sigma = \{enc/2, dec/2\}$
$E = \{dec(enc(x, y), y) \to x\}$
$\Lambda = \{tag, c, k, m\}$

destructors(E) $= \{dec\}$
constructors(E) $= \{enc\}$

It is always the case that a thread with free name tag knows the key.
$P := \Box H key.(@tag \wedge K key \mid T)$

$(\nu k).tag\langle k\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \mid (\nu k).tag\langle k\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \; + \; (\nu k).tag\langle k\rangle.0 \vDash P$

Example

$\Sigma = \{enc/2, dec/2\}$ destructors(E) $= \{dec\}$
$E = \{dec(enc(x, y), y) \to x\}$ constructors(E) $= \{enc\}$
$\Lambda = \{tag, c, k, m\}$

It is always the case that a thread with free name tag knows the key.
$P := \Box H key.(@tag \land K key \mid T)$

$(\nu k).tag\langle k\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \mid (\nu k).tag\langle k\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.c\langle c\rangle.0 \ + \ (\nu k).tag\langle k\rangle.0 \vDash P$

$(\nu k).tag\langle enc(k, c)\rangle.0 \mid (\nu k).tag\langle k\rangle.0 \vDash P$

Attacker

Attacker

- we want to model a *Dolev-Yao* attacker

Attacker

- we want to model a *Dolev-Yao* attacker
- we already have a specific operation for attacker output

Attacker

- we want to model a *Dolev-Yao* attacker
- we already have a specific operation for attacker output
- that can send any message create having all the knowledge that a perfect reasoner would deduce from the message it sees

Attacker

- we want to model a *Dolev-Yao* attacker
- we already have a specific operation for attacker output
- that can send any message create having all the knowledge that a perfect reasoner would deduce from the message it sees
- we assume to have a process that model the protocol of interest P

Attacker

- we want to model a *Dolev-Yao* attacker
- we already have a specific operation for attacker output
- that can send any message create having all the knowledge that a perfect reasoner would deduce from the message it sees
- we assume to have a process that model the protocol of interest P
- the idea is to express a procedure to create an extra process to put in parallel $P \mid E$

Attacker

- we want to model a *Dolev-Yao* attacker
- we already have a specific operation for attacker output
- that can send any message create having all the knowledge that a perfect reasoner would deduce from the message it sees
- we assume to have a process that model the protocol of interest P
- the idea is to express a procedure to create an extra process to put in parallel $P \mid E$
- **the attacker process** E

Attacker

- we want to model a *Dolev-Yao* attacker
- we already have a specific operation for attacker output
- that can send any message create having all the knowledge that a perfect reasoner would deduce from the message it sees
- we assume to have a process that model the protocol of interest P
- the idea is to express a procedure to create an extra process to put in parallel $P \mid E$
- **the attacker process** E
- then we have to prove that the process $P \mid E$ satisfy the logical formula asserting the property of interest

Attacker Generation

Attacker Generation

- ***Dolev-Yao* attacker:** intercept all communications of the principals and be able to inject any message it can produce with knowledge derived until that time

Attacker Generation

- ***Dolev-Yao* attacker:** intercept all communications of the principals and be able to inject any message it can produce with knowledge derived until that time
- E is a process that for all output of P perform an input (storing the received message), and for all input of P performs an attacker output

Attacker Generation

- ***Dolev-Yao* attacker:** intercept all communications of the principals and be able to inject any message it can produce with knowledge derived until that time
- E is a process that for all output of P perform an input (storing the received message), and for all input of P performs an attacker output

Procedure for generating attacker:

```
proc Attacker(P,S){
    if (P -α-> Q ∧ α = input on c) then c⟨∗⟩.Attacker(Q,S)
    if (P -α-> Q ∧ α = output on c) then c(x).Attacker(Q,S∪{x})
    if ( P -α-/-> ) then m⟨x_1,...,x_n⟩ where x_i ∈ S
```

$$\texttt{if } (P \xrightarrow{\alpha} Q \wedge \alpha = \texttt{input on } c) \texttt{ then } c\langle * \rangle\texttt{.Attacker(Q,S)}$$

$$\texttt{if } (P \xrightarrow{\alpha} Q \wedge \alpha = \texttt{output on } c) \texttt{ then } c(x)\texttt{.Attacker(Q,S}\cup\{x\}\texttt{)}$$

$$\texttt{if } (\ P \stackrel{\alpha}{\not\to}\ ) \texttt{ then } m\langle x_1, \ldots, x_n \rangle \texttt{ where } x_i \in S$$

Example

$\Sigma =$
$\{enc/2, dec/2, pair/2, \pi_1/1, \pi_2/1, t/1\}$

$$E = \{dec(enc(x, y), y) \to x$$
$$\pi_1(pair(x, y)) \to x$$
$$\pi_2(pair(x, y)) \to y\}$$

destructors$(E) =$
$\{dec, \pi_1, \pi_2\}$

constructors$(E) =$
$\{enc, pair, t\}$

Example

$\Sigma =$
$\{enc/2, dec/2, pair/2, \pi_1/1, \pi_2/1, t/1\}$

$$E = \{dec(enc(x, y), y) \to x$$
$$\pi_1(pair(x, y)) \to x$$
$$\pi_2(pair(x, y)) \to y\}$$

destructors$(E) =$
$\{dec, \pi_1, \pi_2\}$

constructors$(E) =$
$\{enc, pair, t\}$

We consider the following trivial protocol

$$A \to B : \{key_{ab}, N\}_{key}$$
$$B \to A : \{N - 1\}_{key_{ab}}$$

Example

Protocol modeling:

$$
\begin{aligned}
A(key) &= (\nu key_{ab}, N) \\
&\qquad c\langle enc(pair(key_{ab}, N), key)\rangle. \\
&\qquad c(x).[t(N) = dec(x, key_{ab})].\ ok\langle ok\rangle \\
B(key) &= c(x).\mathtt{let} \\
&\qquad key_{ab} = \pi_1(dec(x, key)) \\
&\qquad N = \pi_2(dec(x, key)) \\
&\qquad\qquad \mathtt{in}\ \ c\langle enc(t(N), key_{ab})\rangle \\
Sys &= (\nu key)(A(key) \mid B(key))
\end{aligned}
$$

Example

Attacker modeling:

$$E = c(x).\ c\langle * \rangle.\ c(y).\ c\langle * \rangle.\ mem\langle c, y \rangle$$
$$World = (Sys \mid E)$$

Example

Attacker modeling:

$$E = c(x).\ c\langle * \rangle.\ c(y).\ c\langle * \rangle.\ mem\langle c, y \rangle$$
$$World = (Sys \mid E)$$

we can see:

Example

Attacker modeling:

$$E = c(x).\ c\langle * \rangle.\ c(y).\ c\langle * \rangle.\ mem\langle c, y \rangle$$
$$World = (Sys \mid E)$$

we can see:

$$World \vDash \neg \Diamond Hkey.(2 \mid (@mem \wedge Kkey))$$

Example

Attacker modeling:

$$E = c(x).\ c\langle * \rangle.\ c(y).\ c\langle * \rangle.\ mem\langle c, y\rangle$$
$$World = (Sys \mid E)$$

we can see:

$$World \vDash \neg \Diamond Hkey.(2 \mid (@mem \wedge Kkey))$$

$$World \vDash \neg \Diamond Hkey.(ok(ok).T \wedge (1|\neg Kkey|@mem))$$

Bibliography

Bibliography

- *H. van Ditmarsch, J.Y. Halpern, W. van der Hoek, B. Kooi*. **Handbook of Epistemic Logic**. 2015 College Publications.
  - Ch1. *H. van Ditmarsch, J.Y. Halpern, W. van der Hoek, B. Kooi*. **An Introduction to Logics of Knowledge and Belief**
  - Ch5. *C. Dixon, C. Nalon, R. Ramanujan*. **Knowledge and Time**
  - Ch6. *Lawrence S. Moss*. **Dynamic Epistemic Logic**
  - Ch12. *R. Pucella*. **Knowledge and Security**

- *D. Palladino, C. Palladino*. **Logiche non classiche, un'introduzione**. 2007 Carocci Editore.

- *B. Toninho, L. Caires*. **A spatial-Epistemic Logic for Reasoning about Security Protocols**. 8th International Workshop on Security Issues in Concurrency (SecCo'10), Paris 2010.

- *B. Toninho*. **A Logic and Tool for Local Reasoning about Security Protocols**. MSc Dissertation (2009).

- *M. Abadi, C. Fournet*. **Mobile values, new names, and secure communication**. POPL'01: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, New York, NY, USA, ACM (2001)