Regular Model Checking of Epistemic Logic

Daniel Stan[1]

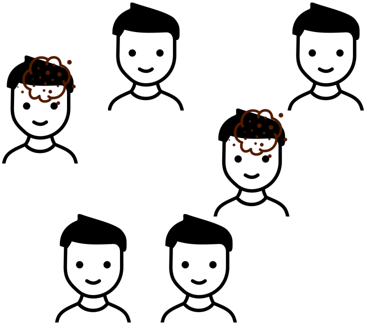
June 27, 2022

1 TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

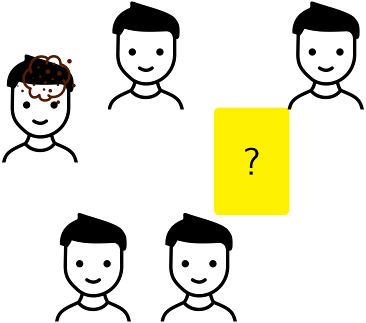
2 MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

joint work with Anthony W. Lin[1,2], Felix Thoma[1]

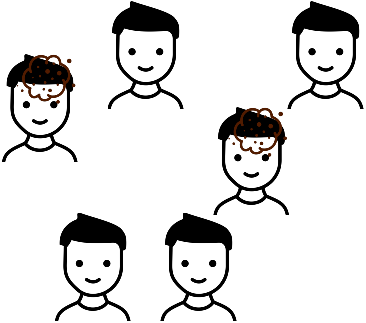
Muddy Children Puzzle (Littlewood, 1953)



Muddy Children Puzzle (Littlewood, 1953)

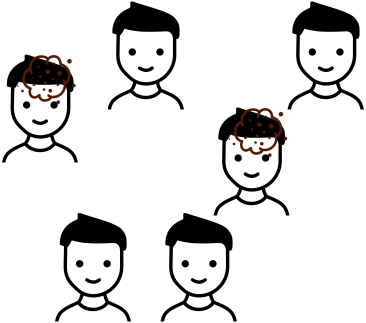


Muddy Children Puzzle (Littlewood, 1953)



Father: at least one of you is muddy!

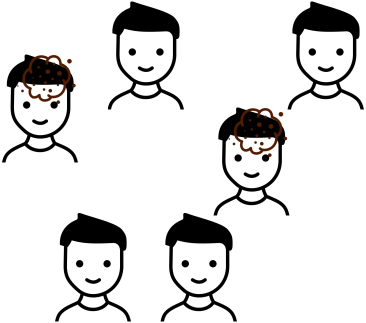
Muddy Children Puzzle (Littlewood, 1953)



Father: at least one of you is muddy!

Children: ???

Muddy Children Puzzle (Littlewood, 1953)

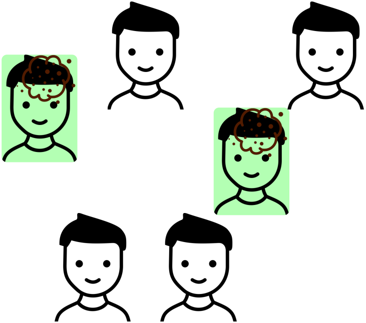


Father: at least one of you is muddy!

Children: ???

Father: indeed, no one knows.

Muddy Children Puzzle (Littlewood, 1953)



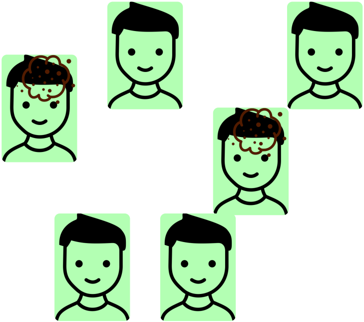
Father: at least one of you is muddy!

Children: ???

Father: indeed, no one knows.

Muddy children: ah, yes we know we're muddy.

Muddy Children Puzzle (Littlewood, 1953)



Father: at least one of you is muddy!

Children: ???

Father: indeed, no one knows.

Muddy children: ah, yes we know we're muddy.

Clean children: ah, yes we know we're clean.

Setting

- **Common Knowledge** framework
- Communication primitive: **Public Announcement**
- **Model checking** problem: given some model $\mathcal{M}$ and some specification φ, decide whether $\mathcal{M} \models \varphi$;
- **Parameterized** problem: the number of agents (children) is **not fixed**.

Setting

- **Common Knowledge** framework
- Communication primitive: **Public Announcement**
- **Model checking** problem: given some model $\mathcal{M}$ and some specification φ, decide whether $\mathcal{M} \models \varphi$;
- **Parameterized** problem: the number of agents (children) is **not fixed**.

Applications: analysis of communication protocols involving identical arbitrarily many processes

Setting

- **Common Knowledge** framework
- Communication primitive: **Public Announcement**
- **Model checking** problem: given some model $\mathcal{M}$ and some specification φ, decide whether $\mathcal{M} \models \varphi$;
- **Parameterized** problem: the number of agents (children) is **not fixed**.

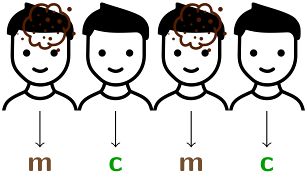
Applications: analysis of communication protocols involving identical arbitrarily many processes

Outline

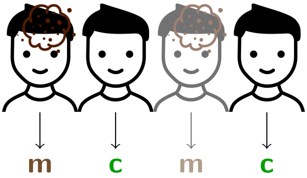
1. **Parameterized Public Announcement Logic** on **Regular Structures**;
2. **Active Learning** of **Iterated Public Announcement**
3. **Extensions**

1: Parameterized Public Announcement Logic (Modal Logic)

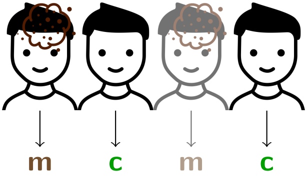
Indistinguishability Relation



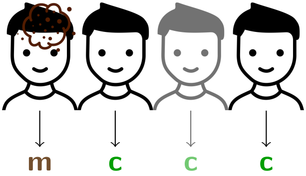
Indistinguishability Relation



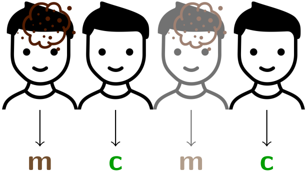
Indistinguishability Relation



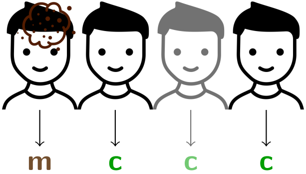
2 might think:



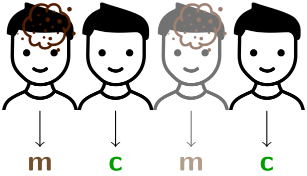
Indistinguishability Relation



2 might think: $\overset{2}{\sim}$

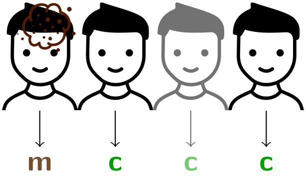


Indistinguishability Relation



2 might think:

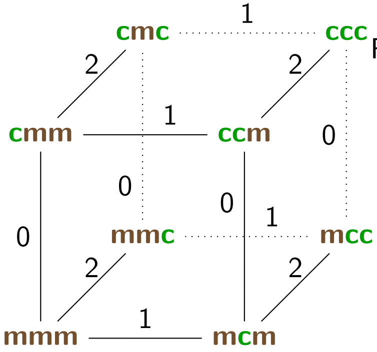
$\overset{2}{\sim}$



(S5): For every i, $\overset{i}{\sim}$ is an equivalence relation.

Kripke Structures

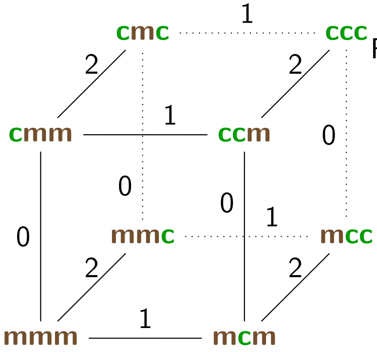
Case with 3 children



From a given state s,

Kripke Structures

Case with 3 children

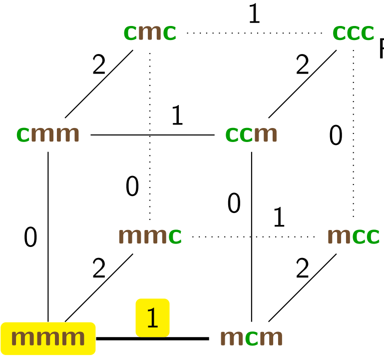


From a given state s,

- If $s \overset{i}{\sim} t$, i *may think* we are in t.

Kripke Structures

Case with 3 children



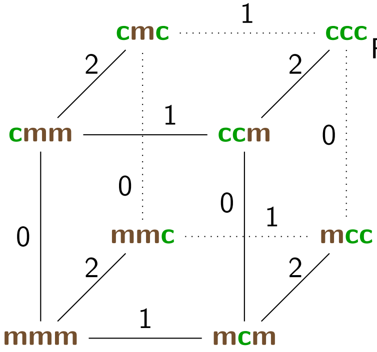
From **mmm**, agent 1 knows that third letter is **m**

From a given state s,

- If $s \overset{i}{\sim} t$, i *may think* we are in t.
- If for all t such that $s \overset{i}{\sim} t$, t satisfies some property φ, then i *knows* that φ holds (from s).

Kripke Structures

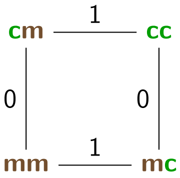
Case with 3 children



From a given state s,

- If $s \overset{i}{\sim} t$, i *may think* we are in t.
- If for all t such that $s \overset{i}{\sim} t$, t satisfies some property φ, then i *knows* that φ holds (from s).
- Any agent knows the structure of the graph: *common knowledge*.

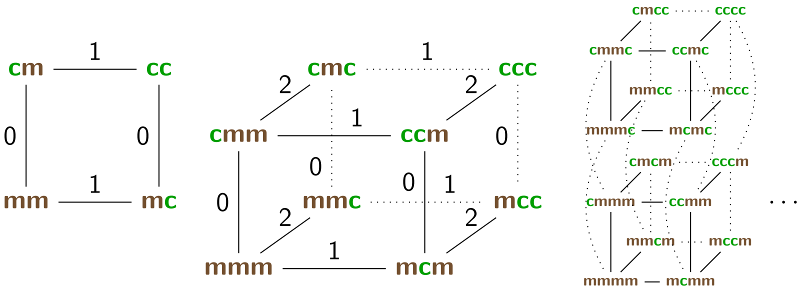
Parameterized Verification of a property φ



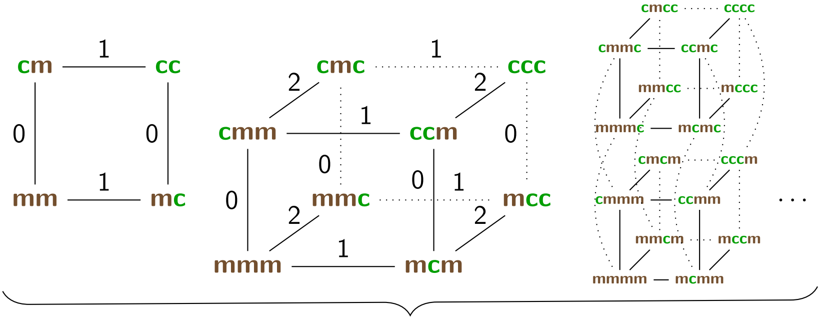
Parameterized Verification of a property φ

$$\begin{tikzcd} \mathbf{cm} \arrow[r, dash, "1"] \arrow[d, dash, "0"'] & \mathbf{cc} \arrow[d, dash, "0"'] \\ \mathbf{mm} \arrow[r, dash, "1"] & \mathbf{mc} \end{tikzcd} \models \varphi?$$

Parameterized Verification of a property φ



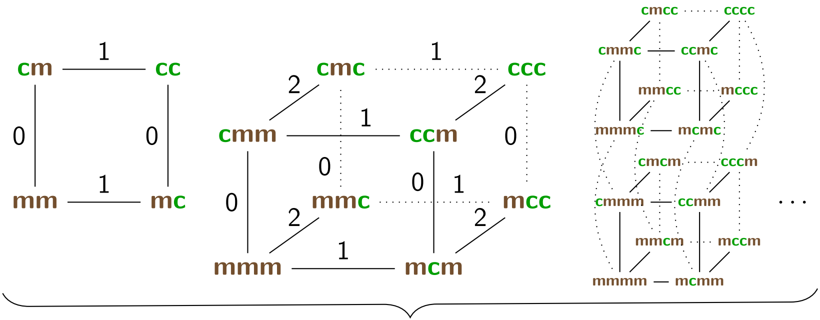
Parameterized Verification of a property φ



$\models \varphi$?

$\hookrightarrow$ **Infinite** collection of systems

Parameterized Verification of a property φ



$\models \varphi$? (for **all** states of **all** instances)

$\hookrightarrow$ **Infinite** collection of systems

PAL: Public Announcement Logic (Plaza, 07)

$\varphi ::= p \mid \top \mid \varphi \wedge \varphi \mid \neg \varphi \mid K_a \varphi \mid \langle \varphi ! \rangle \psi$

Where:
$p \in AP$ is an atomic proposition;
$a \in \mathbb{N}$ is a constant;

PAL: Public Announcement Logic (Plaza, 07)

After announcing φ, ψ holds

a knows φ

$$\varphi ::= p \mid \top \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_a\varphi \mid \langle\varphi!\rangle\psi$$

Where:

$p \in AP$ is an atomic proposition;

$a \in \mathbb{N}$ is a constant;

PPAL: Parameterized Public Announcement Logic

After announcing φ, ψ holds

i knows φ

$$\varphi ::= p_{\mathbf{i}} \mid \top \mid \varphi \wedge \varphi \mid \neg \varphi \mid K_{\mathbf{i}}\varphi \mid \langle \varphi ! \rangle \psi \mid \exists \mathbf{i} : \varphi \mid \mathbf{i} = \mathbf{0} \mid \mathbf{i} \% \mathbf{k} = \mathbf{0} \mid \mathbf{i} = \mathbf{j} + \mathbf{a}$$

Where:

$p \in AP$ is an atomic proposition;

$a \in \mathbb{N}$ is a constant;

i, j are index variables, for **agents** and **atomic propositions**.

PPAL: **Parameterized** Public Announcement Logic

After announcing φ, ψ holds

i knows φ

$$\varphi ::= p_{\mathbf{i}} \mid \top \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_{\mathbf{i}}\varphi \mid \langle \varphi ! \rangle \psi \mid \exists \mathbf{i} : \varphi \mid \mathbf{i} = \mathbf{0} \mid \mathbf{i} \% \mathbf{k} = \mathbf{0} \mid \mathbf{i} = \mathbf{j} + \mathbf{a}$$

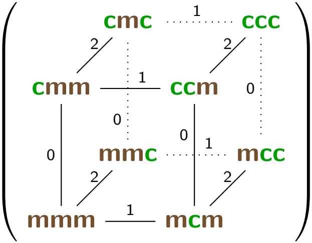
Where:
$p \in AP$ is an atomic proposition;
$a \in \mathbb{N}$ is a constant;
i, j are index variables, for **agents** and **atomic propositions**. Modal logic, also similar to wS1S. Now combined with Public Announcements.

Semantics of a PPAL formula φ

$$
[\![\varphi]\!] \left(\begin{tikzcd}
 & \mathbf{cmc} \arrow[rr, dotted, dash, "1"] \arrow[dd, dotted, dash, "0"'] & & \mathbf{ccc} \arrow[dd, dotted, dash, "0"'] \\
\mathbf{cmm} \arrow[ur, dash, "2"] \arrow[rr, dash, "1"] \arrow[dd, dash, "0"'] & & \mathbf{ccm} \arrow[ur, dash, "2"] \arrow[dd, dash, "0"'] & \\
 & \mathbf{mmc} \arrow[rr, dotted, dash, "1"] & & \mathbf{mcc} \\
\mathbf{mmm} \arrow[ur, dash, "2"] \arrow[rr, dash, "1"] & & \mathbf{mcm} \arrow[ur, dash, "2"] &
\end{tikzcd} \right) = \{ s \mid s \models \varphi \}
$$

Semantics of a PPAL formula φ

"All children are clean"

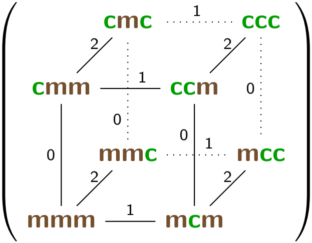
$[\![\varphi]\!]$


$= \{s \mid s \vDash \varphi\}$

Semantics of a PPAL formula φ

"All children are clean"

$[\![\forall i : \neg \mathbf{m}_i]\!]$

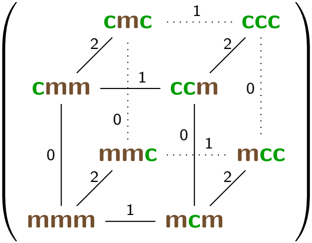


$=$

Semantics of a PPAL formula φ

"All children are clean"

$[\![\forall i : \neg \mathbf{m}_i]\!]$

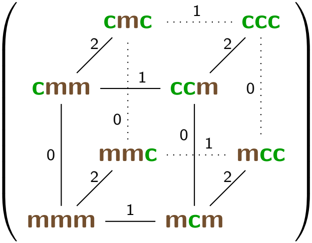


$= \{\mathbf{ccc}\}$

Semantics of a PPAL formula φ

"After announcing there is at least one muddy child,
$\underbrace{\text{all the muddy children know they're muddy''}}_{\varphi \equiv \forall j : \mathbf{m}_j \rightarrow K_j \mathbf{m}_j}$

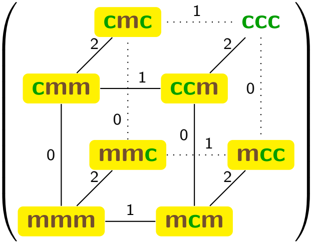
$[\![\langle\ \exists i : \mathbf{m}_i\ !\rangle\varphi]\!]$

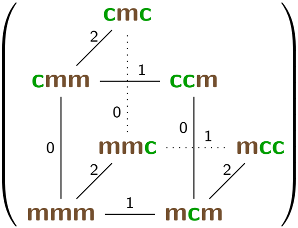


$=$

Semantics of a PPAL formula φ

"After announcing there is at least one muddy child,
$\underbrace{\text{all the muddy children know they're muddy"}}_{\varphi \equiv \forall j : \mathbf{m}_j \to K_j \mathbf{m}_j}$

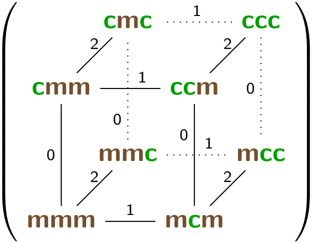
$[\![\langle \boxed{\exists i : \mathbf{m}_i} ! \rangle \varphi]\!]$ 

$= [\![\varphi]\!]$ 

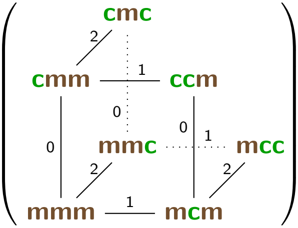
Semantics of a PPAL formula φ

"After announcing there is at least one muddy child,
$\underbrace{\text{all the muddy children know they're muddy''}}_{\varphi \equiv \forall j : \mathbf{m}_j \to K_j \mathbf{m}_j}$

$[\![\langle\, \exists i : \mathbf{m}_i \;!\rangle\varphi]\!]$



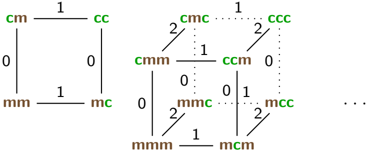
$= [\![\varphi]\!]$



$= \{\mathbf{cmc}, \mathbf{mcc}, \mathbf{ccm}\}$

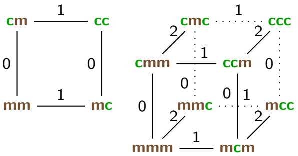
Semantics of φ on a **paramaterized** system

$$[\![\langle \exists i : \mathbf{m}_i! \rangle \forall j : \mathbf{m}_j \to K_j \mathbf{m}_j]\!] \left(\quad \cdots \right) =$$



Semantics of φ on a **paramaterized** system

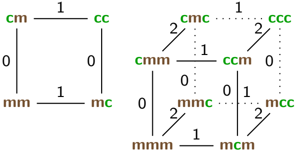
$$[\![\langle \exists i : \mathbf{m}_i! \rangle \forall j : \mathbf{m}_j \to K_j \mathbf{m}_j]\!] \left(\quad \cdots \right) =$$



$$\left\{ \begin{matrix} \mathbf{mc}, \\ \mathbf{cm}, \end{matrix} \qquad \begin{matrix} \mathbf{mcc}, \\ \mathbf{cmc}, \\ \mathbf{ccm}, \end{matrix} \qquad \begin{matrix} \mathbf{mccc}, \\ \mathbf{cmcc}, \\ \mathbf{ccmc}, \\ \mathbf{cccm}, \end{matrix} \ldots \right\}$$

Semantics of φ on a **paramaterized** system

$$[\![\langle \exists i : \mathbf{m}_i ! \rangle \forall j : \mathbf{m}_j \to K_j \mathbf{m}_j]\!] \left(\quad \right. $$

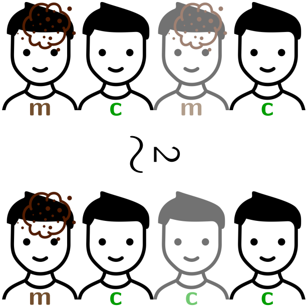


$$\left. \cdots \right) =$$

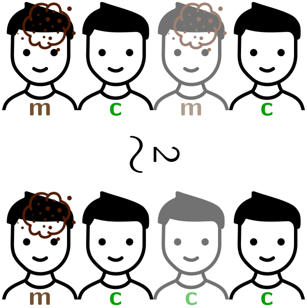
$$\left\{ \begin{array}{cccc} \mathbf{mc}, & \mathbf{mcc}, & \mathbf{mccc}, & \\ \mathbf{cm}, & \mathbf{cmc}, & \mathbf{cmcc}, & \\ & \mathbf{ccm}, & \mathbf{ccmc}, & \\ & & \mathbf{cccm}, & \ldots \end{array} \right\} =$$

$$\{\mathbf{c}\}^* \cdot \{\mathbf{m}\} \cdot \{\mathbf{c}\}^*$$

Regular Encoding



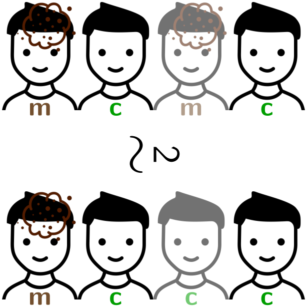
Regular Encoding



Encoded as

m	c	m	c
0	0	1	0
m	c	c	c

Regular Encoding

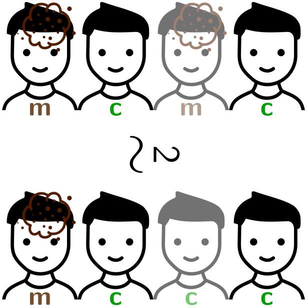


Encoded as

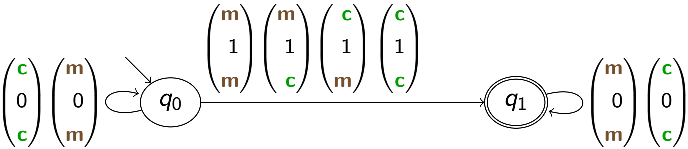
m	**c**	**m**	**c**
0	0	1	0
m	**c**	**c**	**c**

$$\in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0, 1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$

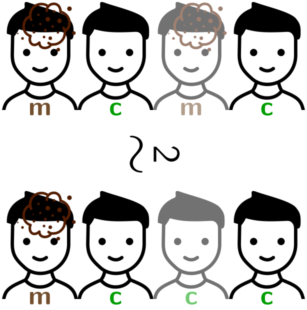
Regular Encoding



Encoded as $\begin{array}{|c|c|c|c|} \hline \mathbf{m} & \mathbf{c} & \mathbf{m} & \mathbf{c} \\ 0 & 0 & 1 & 0 \\ \mathbf{m} & \mathbf{c} & \mathbf{c} & \mathbf{c} \\ \hline \end{array} \in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0, 1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$



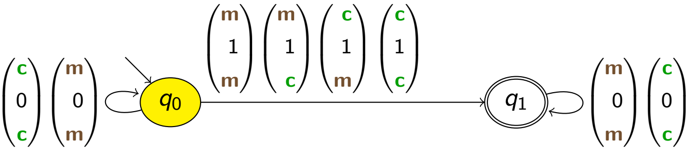
Regular Encoding



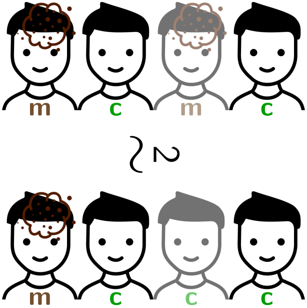
Encoded as

m	**c**	**m**	**c**
0	0	1	0
m	**c**	**c**	**c**

$$\in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0, 1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$



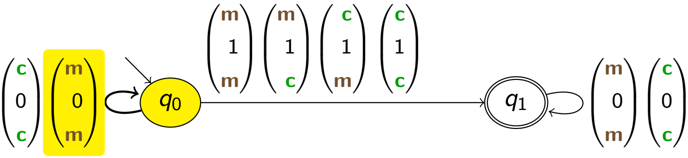
Regular Encoding



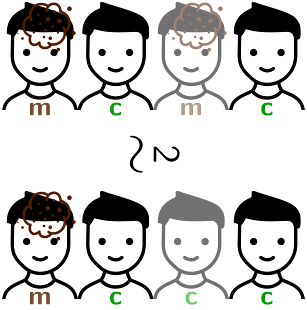
Encoded as

m	c	m	c
0	0	1	0
m	c	c	c

$$\in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0,1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$



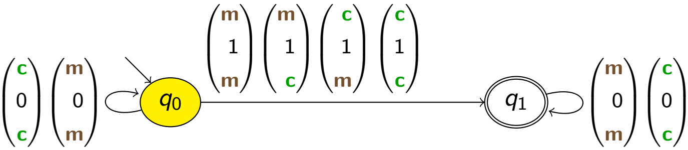
Regular Encoding



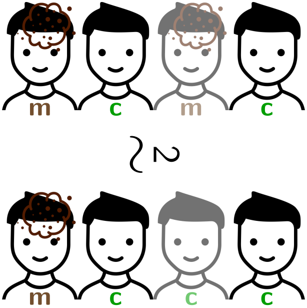
Encoded as

m	c	m	c
0	0	1	0
m	c	c	c

$$\in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0, 1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$



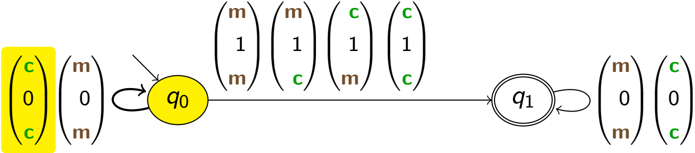
Regular Encoding



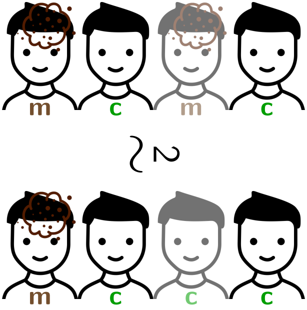
Encoded as

m	c	m	c
0	0	1	0
m	c	c	c

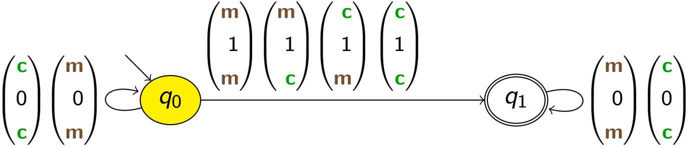
$$\in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0,1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$



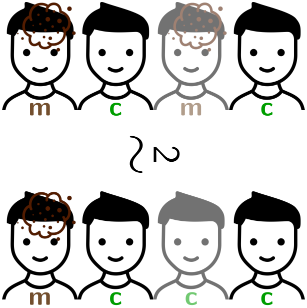
Regular Encoding



Encoded as $\begin{array}{|c|c|c|c|} \hline \mathbf{m} & \mathbf{c} & \mathbf{m} & \mathbf{c} \\ 0 & 0 & 1 & 0 \\ \mathbf{m} & \mathbf{c} & \mathbf{c} & \mathbf{c} \\ \hline \end{array} \in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0, 1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$



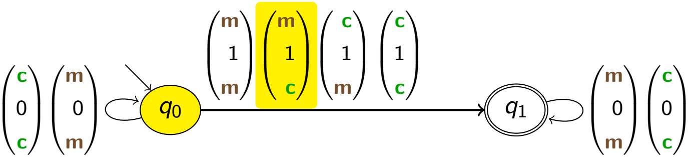
Regular Encoding



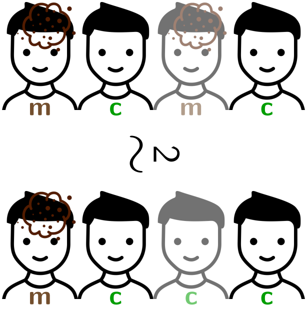
Encoded as

m	c	m	c
0	0	1	0
m	c	c	c

$$\in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0,1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$

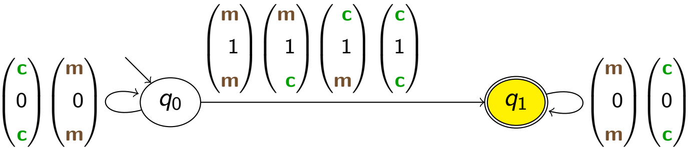


Regular Encoding

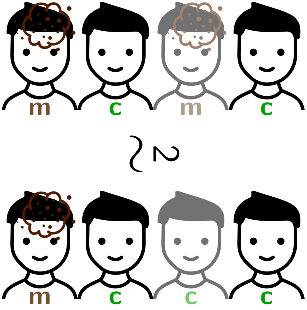


Encoded as

$$\begin{array}{|c|c|c|c|} m & c & m & c \\ 0 & 0 & 1 & 0 \\ m & c & c & c \end{array} \in \begin{pmatrix} \{m, c\} \\ \times \\ \{0,1\} \\ \times \\ \{m, c\} \end{pmatrix}^*$$



Regular Encoding



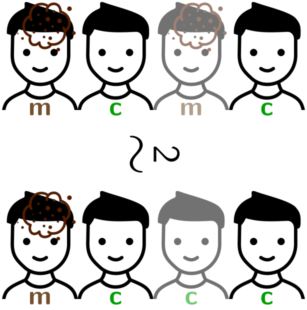
Encoded as

m	c	m	c
0	0	1	0
m	c	c	c

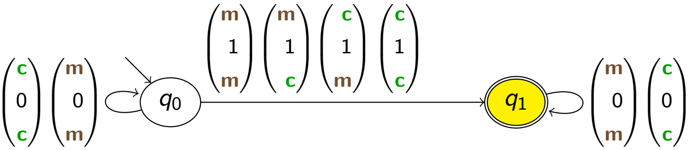
$$\in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0,1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$

$$\begin{tikzcd} q_0 \arrow[loop left, "\begin{pmatrix} \mathbf{c} \\ 0 \\ \mathbf{c} \end{pmatrix} \begin{pmatrix} \mathbf{m} \\ 0 \\ \mathbf{m} \end{pmatrix}"] \arrow[rrr, "\begin{pmatrix} \mathbf{m} \\ 1 \\ \mathbf{m} \end{pmatrix} \begin{pmatrix} \mathbf{m} \\ 1 \\ \mathbf{c} \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ 1 \\ \mathbf{m} \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ 1 \\ \mathbf{c} \end{pmatrix}"] & & & q_1 \arrow[loop right, "\begin{pmatrix} \mathbf{m} \\ 0 \\ \mathbf{m} \end{pmatrix} \boxed{\begin{pmatrix} \mathbf{c} \\ 0 \\ \mathbf{c} \end{pmatrix}}"] \end{tikzcd}$$

Regular Encoding



$$\text{Encoded as } \begin{array}{|c|c|c|c|}\hline \mathbf{m} & \mathbf{c} & \mathbf{m} & \mathbf{c} \\ 0 & 0 & 1 & 0 \\ \mathbf{m} & \mathbf{c} & \mathbf{c} & \mathbf{c} \\ \hline \end{array} \in \begin{pmatrix} \{\mathbf{m}, \mathbf{c}\} \\ \times \\ \{0,1\} \\ \times \\ \{\mathbf{m}, \mathbf{c}\} \end{pmatrix}^*$$



Definition (**Regular** Kripke structure)

$\mathcal{M} = (S, \Sigma, AP, \sim, L)$ where:

- Σ finite alphabet;
- AP finite set of atomic propositions;
- $S \subseteq \Sigma^*$;
- For all $0 \leq i < |s|$, $L_i(s) \subseteq AP$;
- $\sim$ is encoded as a **length-preserving** transducer:

$$T_{\mathcal{M}} = \left\{ s \otimes \left(\underbrace{0 \ldots 0}_{i} \cdot 1 \cdot \underbrace{0 \ldots 0}_{|s|-i-1} \right) \otimes t \;\middle|\; s \overset{i}{\sim} t \right\}$$

Contribution: Regular Semantics of a PPAL formula

$$\left\{ \begin{array}{|c|}\hline w_1 \\ \hline 0^{i-1}10^{n-i-1} \\ \hline w_2 \\ \hline \end{array} \;\middle|\; w_1 \overset{i}{\sim} w_2 \right\} \xrightarrow{[\![\varphi]\!]} \left\{ \boxed{\;w\;} \;\middle|\; \mathcal{M}, w \models \varphi \right\}$$

Contribution: Regular Semantics of a PPAL formula

$$\left\{ \begin{array}{|c|}\hline w_1 \\ \hline 0^{i-1}10^{n-i-1} \\ \hline w_2 \\ \hline \end{array} \;\middle|\; w_1 \overset{i}{\sim} w_2 \right\} \xrightarrow{[\![\varphi]\!]} \left\{ \boxed{w} \;\middle|\; \mathcal{M}, w \models \varphi \right\}$$

Theorem

If $\mathcal{M}$ is a regular Kripke structure, then $[\![\varphi]\!](\mathcal{M})$ is a **regular language***.*
Moreover, the transformation is **effective***.*

PPAL model checking is **decidable**.

Contribution: Regular Semantics of a PPAL formula

$$\left\{ \begin{array}{c} \boxed{x} \\ + \\ \boxed{w_1} \\ \boxed{0^{i-1}10^{n-i-1}} \\ \boxed{w_2} \end{array} \middle| \; w_1 \overset{i}{\sim}_x w_2 \right\} \xrightarrow{[\![\varphi]\!]} \left\{ \begin{array}{c} \boxed{x} \\ + \\ \boxed{w} \end{array} \middle| \; \mathcal{M}_x, w \vDash \varphi \right\}$$

Theorem

If $\mathcal{M}$ is a regular Kripke structure, then $[\![\varphi]\!](\mathcal{M})$ is a **regular language***.*
Moreover, the transformation is **uniformly effective***.*

PPAL model checking is **decidable**.

Contribution: Regular Semantics of a PPAL formula

$$\left\{ \begin{array}{c} \boxed{x} \\ + \\ \boxed{w_1} \\ \boxed{0^{i-1}10^{n-i-1}} \\ \boxed{w_2} \end{array} \middle| \; w_1 \overset{i}{\sim}_x w_2 \right\} \xrightarrow{[\![\varphi]\!]} \left\{ \begin{array}{c} \boxed{x} \\ + \\ \boxed{w} \end{array} \middle| \; \mathcal{M}_x, w \vDash \varphi \right\}$$

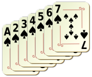
Theorem

If $\mathcal{M}$ is a regular Kripke structure, then $[\![\varphi]\!](\mathcal{M})$ is a **regular language***.*
Moreover, the transformation is **uniformly** **effective***.*

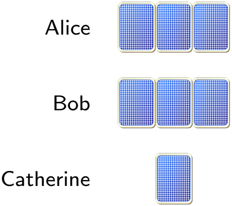
PPAL model checking is **decidable**.

Application Verify the parameterized solution of van Ditmarsch (2003) for 3+x+1 cards.

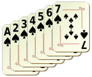
Another Example: Russian card problem, with $3 + 3 + 1$

cards: 

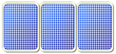
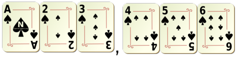
Alice's goal:
- Making Bob aware of her hand;
- Not disclosing any card to Catherine (except her own).

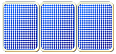


Another Example: Russian card problem, with $3 + 3 + 1$

cards: 

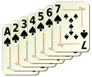
Alice's goal:
- Making Bob aware of her hand;
- Not disclosing any card to Catherine (except her own).

Alice  "I have one of these hands:" 

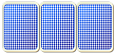
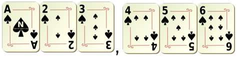
Bob 

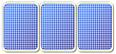
Catherine 

Another Example: Russian card problem, with $3+3+1$

cards: 

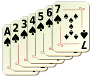
Alice's goal:
✓ Making Bob aware of her hand;
• Not disclosing any card to Catherine (except her own).

Alice  "I have one of these hands:" 

Bob 

Catherine 

Another Example: Russian card problem, with $3 + 3 + 1$

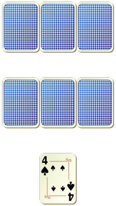
cards: 

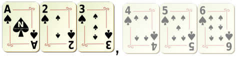
Alice's goal:
✓ Making Bob aware of her hand;
✗ Not disclosing any card to Catherine (except her own).

Alice

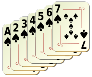
Bob

Catherine



"I have one of these hands:" 

Another Example: Russian card problem, with $3 + 3 + 1$

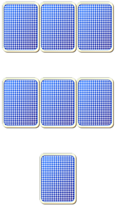
cards: 

Alice's goal:
- Making Bob aware of her hand;
- Not disclosing any card to Catherine (except her own).

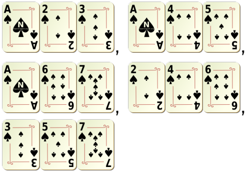
Alice

Bob

Catherine

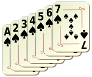


"I have one of these hands:"



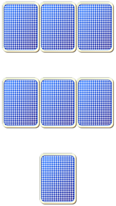
Another Example: Russian card problem, with $3 + 3 + 1$

cards:

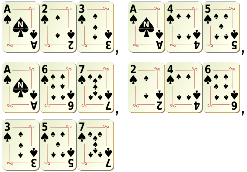


Alice's goal:
✓ Making Bob aware of her hand;
✓ Not disclosing any card to Catherine (except her own).

Alice
Bob
Catherine



"I have one of these hands:"



2: Iterated Announcement

- Safety Regular Model Checking
- Active Learning approach
- Learning Disappearance Relation

How many announcements? Iterated Announcement

How many announcements before a (sub)formula holds?
Muddy Children Example:
One muddy child $\rightarrow$ 1 PA;

How many announcements? Iterated Announcement

How many announcements before a (sub)formula holds?

Muddy Children Example:

One muddy child $\rightarrow$ 1 PA;
Two muddy children $\rightarrow$ 2 PA;
Three muddy children $\rightarrow$ 3 PA $\ldots$.

How many announcements? Iterated Announcement

How many announcements before a (sub)formula holds?

Muddy Children Example:

One muddy child $\rightarrow$ 1 PA;

Two muddy children $\rightarrow$ 2 PA;

Three muddy children $\rightarrow$ 3 PA . . . .

k muddy children require **k** announcements to conclude on their state.

How many announcements? Iterated Announcement

How many announcements before a (sub)formula holds?

Muddy Children Example:

One muddy child $\rightarrow$ 1 PA;

Two muddy children $\rightarrow$ 2 PA;

Three muddy children $\rightarrow$ 3 PA ....

k muddy children require **k** announcements to conclude on their state.

No **fixed** PPAL formula

How many announcements? Iterated Announcement

How many announcements before a (sub)formula holds?
Muddy Children Example:
One muddy child $\rightarrow$ 1 PA;
Two muddy children $\rightarrow$ 2 PA;
Three muddy children $\rightarrow$ 3 PA ....
k muddy children require **k** announcements to conclude on their state.
No **fixed** PPAL formula
$PPAL^* = PPAL +$ **iterated announcement**:

$$\langle\varphi!\rangle^*\psi \equiv \exists k \in \mathbb{N} : \underbrace{\langle\varphi!\rangle \ldots \langle\varphi!\rangle}_{k \text{ times}} \psi$$

Safety Regular Model Checking

Let us first consider a more classical problem:

Safety Regular Model Checking

Let us first consider a more classical problem:

Definition (Safety Analysis)

Given $\mathcal{M} = (S, \Sigma, AP, \sim, L)$ a regular Kripke structure and two regular sets $Init, Bad \in \mathrm{Reg}(\Sigma)$.
Is the system *Safe*? That is to say:

$$\forall w \in Init, \forall w' \in \Sigma^*, w \sim^* w' \quad \Rightarrow w' \notin Bad$$

Safety Regular Model Checking

Let us first consider a more classical problem:

Definition (Safety Analysis)

Given $\mathcal{M} = (S, \Sigma, AP, \sim, L)$ a regular Kripke structure and two regular sets $Init, Bad \in \mathrm{Reg}(\Sigma)$.
Is the system *Safe*? That is to say:

$$\forall w \in Init, \forall w' \in \Sigma^*, w \sim^* w' \quad \Rightarrow w' \notin Bad$$

In other words: **Decide** whether $(Init \otimes Bad) \cap T_{\sim^*} = \emptyset$, where $T_{\sim^*}$ is the transducer of the **transisitive closure** of $\sim$.
Decidability:

Safety Regular Model Checking

Let us first consider a more classical problem:

Definition (Safety Analysis)

Given $\mathcal{M} = (S, \Sigma, AP, \sim, L)$ a regular Kripke structure and two regular sets $\mathit{Init}, \mathit{Bad} \in \mathrm{Reg}(\Sigma)$.
Is the system *Safe*? That is to say:

$$\forall w \in \mathit{Init}, \forall w' \in \Sigma^*, w \sim^* w' \quad \Rightarrow w' \notin \mathit{Bad}$$

In other words: **Decide** whether $(\mathit{Init} \otimes \mathit{Bad}) \cap T_{\sim^*} = \emptyset$, where $T_{\sim^*}$ is the transducer of the **transisitive closure** of $\sim$.
Decidability:**No**

Safety Analysis Strategies

Some techniques:

- Regular Model Checking **Using Inference** of Regular Languages (Habermehl and Vojnar, 04)
- Parameterized verification through **view abstraction** (Abdulla, Haziza, and Holik, 15)
- Regular Model Checking using **Widening Techniques** (Touili, 01)
- Regular Model Checking Using **Solver Technologies** and **Automata Learning** (Neider and Jansen, 13)

Safety Analysis Strategies

Some techniques:

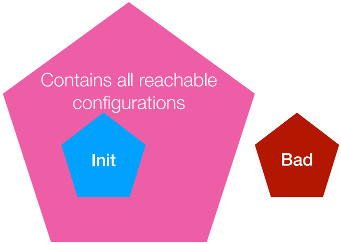
- Regular Model Checking **Using Inference** of Regular Languages (Habermehl and Vojnar, 04)
- Parameterized verification through **view abstraction** (Abdulla, Haziza, and Holik, 15)
- Regular Model Checking using **Widening Techniques** (Touili, 01)
- Regular Model Checking Using **Solver Technologies** and **Automata Learning** (Neider and Jansen, 13)

Most of these are based on finding a **regular invariant**:

Definition

$I \in \mathrm{Reg}(\Sigma)$ such that

1. $\mathit{Init} \subseteq I$;
2. $I \cap \mathit{Bad} = \emptyset$;
3. $\mathit{Post}(I) \subseteq I$



Active Learning: the 20 questions game example

"Think of a character, object or animal, and let me ask you questions."

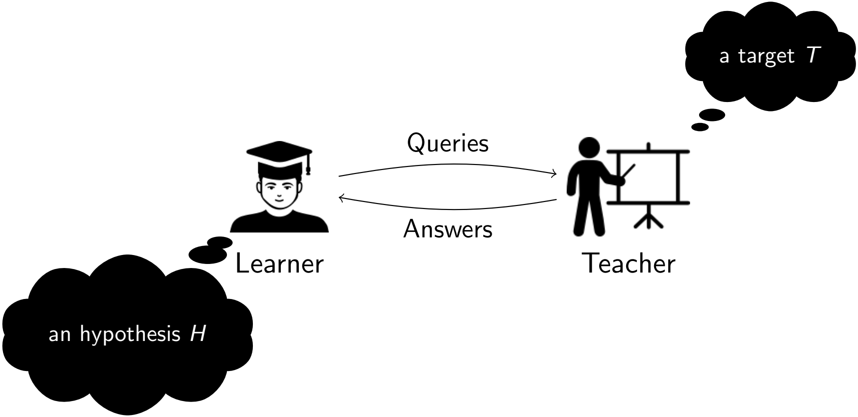
The **learner** asks arbitrary questions

The **teacher** answers by `YES`/`NO`/`MAYBE`.

`https://akinator.com`

`https://en.wikipedia.org/wiki/Twenty_questions`

Active Machine Learning



The **Concept class** $\mathcal{C}$ describes all possible values of $H \in \mathcal{C}$;
Goal for Learner: find $H = T$ or at least $H \sim= T$;
Active learning: the learner chooses the questions.

Regular Language Active Learning (Angluin's L*, 87)

For regular machine learning, the concept to learn is a **finite automaton** $\mathcal{H}$:

- **concept class** $\mathcal{C}$ is the set of all finite automata over Σ
- The **target** is a language $L \subseteq \Sigma^*$.

$$\textbf{Goal}: \mathcal{L}(\mathcal{H}) = L$$

Two types of queries:

- **M**embership queries: "Does $w \in L$?" for some given $w \in \Sigma^*$
  Answer: `YES` or `NO`;
- **EQ**uivalence queries: "Is $\mathcal{L}(\mathcal{H}) = L$?" for some given DFA $\mathcal{H}$
  Answer: `YES` or `NO` and a **counterexample** $w \in \mathcal{L}(\mathcal{H}) \Delta L$

Symmetric Difference of A **and** B: $A \Delta B := A \backslash B \cup B \backslash A$.

Example of a Learner: learnlib

Java Library for active learning of regular languages:
`https://learnlib.de/`

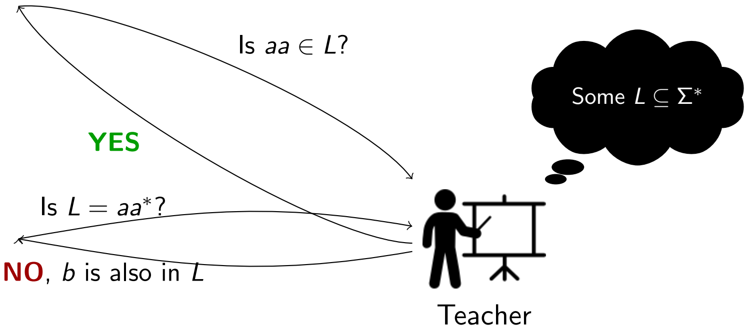
github.com/LearnLib/learnlib/wiki/Instantiating-a-simple-learning-setup

This whole procedure can be implemented as follows:

```java
DefaultQuery<Input, Word<String>> counterexample = null;
do {
    if (counterexample == null) {
        learner.startLearning();
    } else {
        boolean refined = learner.refineHypothesis(counterexample);
        if (!refined) {
            System.err.println("No refinement effected by counterexample!");
        }
    }

    counterexample = eqoracle.findCounterExample(learner.getHypothesisModel(), alphabet);

} while (counterexample != null);

// from here on learner.getHypothesisModel() will provide an accurate model
```

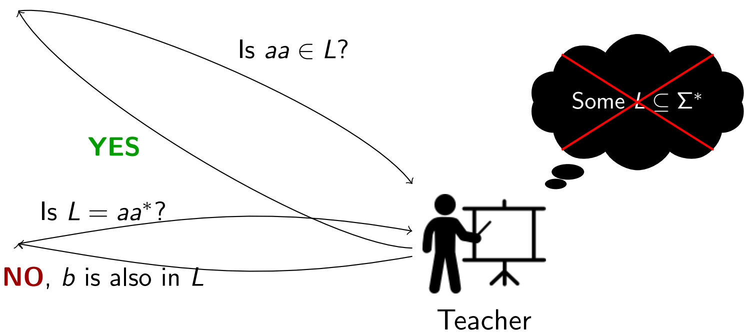
The do-while loop will be executed as long as counterexamples are discovered by the equivalence oracle. Once the loop terminates the hypothesis model provided by the learner is guaranteed to be an exact representation of the target system if the equivalence oracle is guaranteed to find any behavioral mismatches between the hypothesis and the target system (which is the case in this example).

Implementation of a Teacher



A teacher provides **Oracles** (here: a **M** oracle and a **EQ** oracle).

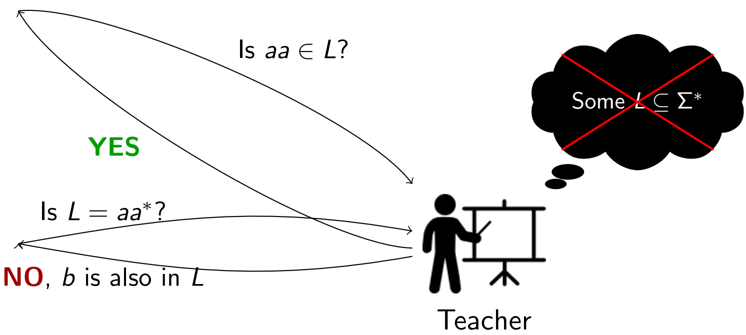
Implementation of a Teacher



A teacher provides **Oracles** (here: a **M** oracle and a **EQ** oracle).

- In practice, the teacher may **not know the target** L, this is fine as long as he **can answers the queries**.

Implementation of a Teacher



A teacher provides **Oracles** (here: a **M** oracle and a **EQ** oracle).

- In practice, the teacher may **not know the target** L, this is fine as long as he **can answers the queries**.
- The target *might* not be **regular**. In this case, the learner will **never manage** to find a suitable automaton $\mathcal{H}$.

A safety teacher

A regular invariant is defined as first-order properties.

Definition (Regular Invariant)

$I \in \mathrm{Reg}(\Sigma)$ such that

1. $\mathit{Init} \subseteq I$;
2. $I \cap \mathit{Bad} = \emptyset$;
3. $\mathit{Post}(I) \subseteq I$

A safety teacher

A regular invariant is defined as first-order properties.

Definition (Regular Invariant)

$I \in \mathrm{Reg}(\Sigma)$ such that

1. $\mathit{Init} \subseteq I$;
2. $I \cap \mathit{Bad} = \emptyset$;
3. $\mathit{Post}(I) \subseteq I$ or equivalently: $(I \otimes \Sigma^*) \cap T_{\sim} \subseteq \Sigma^* \otimes I$

A safety teacher

A regular invariant is defined as first-order properties.

Definition (Regular Invariant)

$I \in \mathrm{Reg}(\Sigma)$ such that

1. $Init \subseteq I$;
2. $I \cap Bad = \emptyset$;
3. $Post(I) \subseteq I$ or equivalently: $(I \otimes \Sigma^*) \cap T_\sim \subseteq \Sigma^* \otimes I$

Oracle for learning some target I:

- **M**embership: Given w, check whether $\exists w' \in Init : w' \sim^* w$.
- **EQ**uivalence queries: Check that the three above properties hold for the hypothesis.

Consequence: using Angluin's L* algorithm, the learning procedure terminates, iff $Post^*_\sim(Init)$ is regular, or contain a bad state.

A safety teacher

A regular invariant is defined as first-order properties.

Definition (Regular Invariant)

$I \in \mathrm{Reg}(\Sigma)$ such that

1. $Init \subseteq I$;
2. $I \cap Bad = \emptyset$;
3. $Post(I) \subseteq I$ or equivalently: $(I \otimes \Sigma^*) \cap T_\sim \subseteq \Sigma^* \otimes I$

Oracle for learning some target I:

- **M**embership: Given w, check whether $\exists w' \in Init : w' \sim^* w$.
- **EQ**uivalence queries: Check that the three above properties hold for the hypothesis.

Consequence: using Angluin's L* algorithm, the learning procedure terminates, iff $Post^*_\sim(Init)$ is regular, or contain a bad state.

Uncovered here: there might be more than one invariant, so there is some slack in the oracle's answer (for membership queries, and for counterexample in (3)).

How many announcements? Iterated Announcement

Muddy Children Example: **k** muddy children require **k** announcements to conclude on their state.

How many announcements? Iterated Announcement

Muddy Children Example: **k** muddy children require **k** announcements to conclude on their state.
No **fixed** PPAL formula

How many announcements? Iterated Announcement

Muddy Children Example: **k** muddy children require **k** announcements to conclude on their state.
No **fixed** PPAL formula
$PPAL^* = PPAL +$ **iterated announcement**:

$$\langle \varphi ! \rangle^* \psi \equiv \exists k \in \mathbb{N} : \underbrace{\langle \varphi ! \rangle \ldots \langle \varphi ! \rangle}_{k \text{ times}} \psi$$

How many announcements? Iterated Announcement

Muddy Children Example: **k** muddy children require **k** announcements to conclude on their state.
No **fixed** PPAL formula
$PPAL^* = PPAL +$ **iterated announcement**:

$$\langle \varphi! \rangle^* \psi \equiv \exists k \in \mathbb{N} : \underbrace{\langle \varphi! \rangle \ldots \langle \varphi! \rangle}_{k \text{ times}} \psi$$

Theorem

Model checking of a regular Kripke structure against:

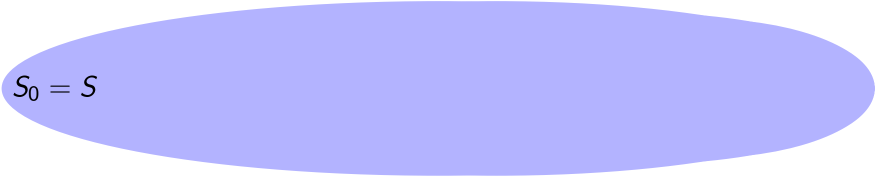
- *a PPAL formula is* **decidable***;*
- *a $PPAL^*$ formula is* **undecidable***.*

We design a **semi-decision** procedure.

Disappearance relation for φ

$$s \preceq t \quad \text{if, and only if,} \quad \forall k,\ s \in S_k \Rightarrow t \in S_k$$

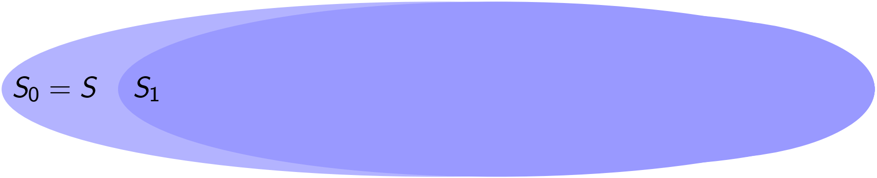
where S_k state space left after k announcements $\langle \varphi ! \rangle$.



Disappearance relation for φ

$$s \preceq t \quad \text{if, and only if,} \quad \forall k,\ s \in S_k \Rightarrow t \in S_k$$

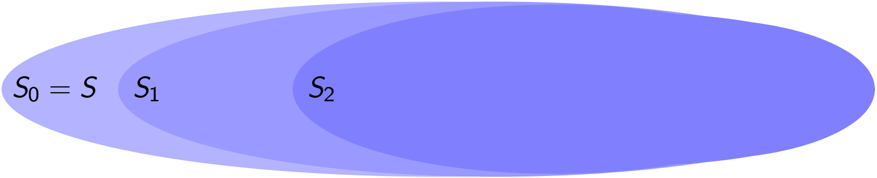
where S_k state space left after k announcements $\langle \varphi! \rangle$.



Disappearance relation for φ

$$s \preceq t \quad \text{if, and only if,} \quad \forall k,\ s \in S_k \Rightarrow t \in S_k$$

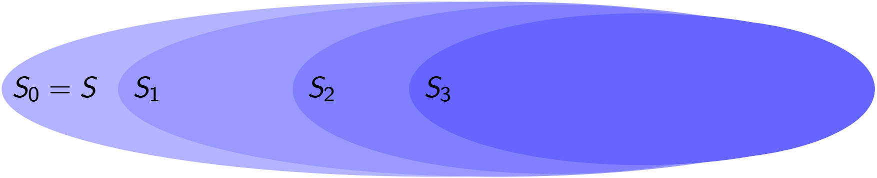
where S_k state space left after k announcements $\langle\varphi!\rangle$.



Disappearance relation for φ

$$s \preceq t \quad \text{if, and only if,} \quad \forall k,\ s \in S_k \Rightarrow t \in S_k$$

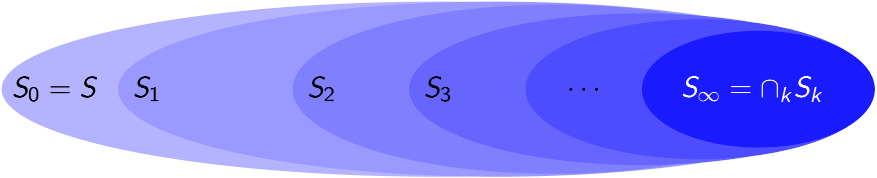
where S_k state space left after k announcements $\langle \varphi ! \rangle$.



Disappearance relation for φ

$$s \preceq t \quad \text{if, and only if,} \quad \forall k,\ s \in S_k \Rightarrow t \in S_k$$

where S_k state space left after k announcements $\langle \varphi ! \rangle$.



Abstracting away from the iteration

Claim: $\mathcal{M}, s \vDash \langle\varphi!\rangle^*\psi$ if, and only if,

$$\underbrace{\exists t \notin S_\infty \wedge t \preceq s \wedge}_{\exists k \in \mathbb{N}} \quad \mathcal{M}_{|\underbrace{\{u \mid t \preceq u\}}_{S_k}}, s \vDash \psi$$

Abstracting away from the iteration

Claim: $\mathcal{M}, s \vDash \langle\varphi!\rangle^*\psi$ if, and only if,

$$\underbrace{\exists t \notin S_\infty \wedge t \preceq s \wedge}_{\exists k \in \mathbb{N}} \quad \mathcal{M}_{|\underbrace{\{u \mid t \preceq u\}}_{S_k}}, s \vDash \psi$$

Consequence: if $\mathcal{M}$ and $L_\preceq = \left\{ \begin{array}{|c|}\hline s \\ \hline t \\ \hline \end{array} \;\middle|\; s \preceq t \right\}$ are **regular**,

then $[\![\langle\varphi!\rangle^*\psi]\!](\mathcal{M})$ is **effectively regular**.

Contribution: Learning $\preceq$

Theorem

Given some PPAL formula φ, and $\preceq$ its disappearance relation. The learning procedure terminates and returns $L_{\preceq}$ if, and only if, $L_{\preceq}$ is **regular**.

Contribution: Learning $\preceq$

Theorem

*Given some PPAL formula φ, and $\preceq$ its disappearance relation. The learning procedure terminates and returns $L_{\preceq}$ if, and only if, $L_{\preceq}$ is **regular**.*

We run L* algorithm (Angluin, 87) by implementing:

1. A **membership oracle**: given $s, t \in S$, $s \overset{?}{\preceq} t$;
2. An **equivalence oracle**: given L' regular, does $L' = L_{\preceq}$ and if not, provide **counterexample** $w \in L \backslash L_{\preceq} \cup L_{\preceq} \backslash L$.

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, *iff:*

1. **Reflexive**
   and
2. **Transitive**
   and
3. **Total**
   and
4. **Some "F" property**

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, *iff:*

1. $\forall s : (s, s) \in R$
   and
2. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R \wedge (s_2, s_3) \in R \rightarrow (s_1, s_3) \in R$
   and
3. $\forall s, t : (s, t) \in R \vee (t, s) \in R$
   and
4. **Some "*F*" property**

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, *iff:*

1. $\forall s : (s, s) \in R$
   and
2. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R \wedge (s_2, s_3) \in R \rightarrow (s_1, s_3) \in R$
   and
3. $\forall s, t : (s, t) \in R \vee (t, s) \in R$
   and
4. **Some "F" property**

All quantifications are made over $\Sigma^k \cap S$ *for some fixed length* k.

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, *iff:*

1. $\forall s : (s, s) \in R$
   and
2. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R$ $(s_1, s_3) \in R$
   and
3. $\forall s, t : (s, t) \in R \vee (t, s) \in R$
   and
4. **Some "F" property**

FO properties over R

All quantifications are made over $\Sigma^k \cap S$ *for some fixed length* k.

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, iff:

1. $\left\{ \begin{bmatrix} w \\ w \end{bmatrix} \middle| \, w \in L_S \right\} \subseteq L_R$

FO properties over R

and

2. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R \ldots (s_1, s_3) \in R$

and

3. $\forall s, t : (s, t) \in R \vee (t, s) \in R$

and

4. **Some "F" property**

All quantifications are made over $\Sigma^k \cap S$ for some fixed length k.

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, iff:

1. $\forall s : (s, s) \in$ ...

   and
2. $\forall s_1, s_2, s_3 : (s_1, s_2) \in$ ... $(s_1, s_3) \in R$

   and
3. $\forall s, t : (s, t) \in R \vee (t, s) \in R$

   and
4. **Some "F" property**

All quantifications are made over $\Sigma^k \cap S$ for some fixed length k.

FO properties over R

$\rightarrow$ **Regular Language Queries**

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, iff:

$\ldots$

4. **Some "F" property**

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, iff:

$\ldots$

4. **Some "*F*" property**

where $sR\cdot = \{u \mid (s,u) \in R\}$ is the set of all states (presumably) not deleted when s **is about to disappear***.*

Two cases:

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, *iff:*

$\dots$

4. **Some "*F*" property**

where $sR\cdot = \{u \mid (s,u) \in R\}$ *is the set of all states (presumably) not deleted when* s **is about to disappear**.

Two cases:

(1) s **really** *disappears;*

(2) s **never** *disappears.*

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, iff:

$\dots$

4. $\forall s \begin{cases} \textit{either } (1)\ \forall t : (s,t) \in R \to (t,s) \notin R \leftrightarrow t \in F(sR\cdot) \\ \quad\ \textit{or } (2)\ \forall t : (s,t) \in R \to (t,s) \in R \land t \in F(sR\cdot) \end{cases}$

where $sR\cdot = \{u \mid (s,u) \in R\}$ is the set of all states (presumably) not deleted when s **is about to disappear***.*

Two cases:
(1) *s* **really** *disappears;*
(2) *s* **never** *disappears.*

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, *iff:*

$\ldots$

4. **FO Property over** R **and** $\{(\mathbf{u}, \mathbf{v}) \mid \mathbf{v} \in \mathbf{F}(\mathbf{uR}\cdot)\}$

where $sR\cdot = \{u \mid (s, u) \in R\}$ *is the set of all states (presumably) not deleted when* s **is about to disappear**.

Two cases:
(1) s **really** *disappears;*
(2) s **never** *disappears.*

Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R = \preceq$, iff:

$\dots$

4. **FO Property over R and $\{(\mathbf{u}, \mathbf{v}) \mid \mathbf{v} \in \mathbf{F}(\mathbf{uR}\cdot)\}$**

   $\rightarrow$ Effective and Uniformly Regular

where $sR\cdot = \{u \mid (s, u) \in R\}$ is the set of all states (presumably) not deleted when s **is about to disappear**.

Two cases:

(1) *s* **really** *disappears;*

(2) *s* **never** *disappears.*

3: Applications and extensions

Muddy Children

"After **arbitrary** but **finitely** many rounds, all the muddy children know they're muddy."

$$\varphi = \langle \exists i : \mathbf{m}_i ! \rangle \langle \underbrace{\exists i : \mathbf{m}_i \wedge \neg K_i \mathbf{m}_i}_{\psi} ! \rangle^* \bot$$

Muddy Children

"After **arbitrary** but **finitely** many rounds, all the muddy children know they're muddy."

$$\varphi = \langle \exists i : \mathbf{m}_i ! \rangle \langle \underbrace{\exists i : \mathbf{m}_i \wedge \neg K_i \mathbf{m}_i !}_{\psi} \rangle^* \bot$$

$$L_{\preceq} = \left\{ \begin{array}{|c|} \hline \mathsf{s} \\ \hline \mathsf{t} \\ \hline \end{array} \middle| \, |s|_{\mathbf{m}} \leq |t|_{\mathbf{m}} \right\} \subseteq (\{\mathbf{m}, \mathbf{c}\} \times \{\mathbf{m}, \mathbf{c}\})^*$$

Muddy Children

"After **arbitrary** but **finitely** many rounds, all the muddy children know they're muddy."

$$\varphi = \langle \exists i : \mathbf{m}_i ! \rangle \langle \underbrace{\exists i : \mathbf{m}_i \wedge \neg K_i \mathbf{m}_i}_{\psi} ! \rangle^* \bot$$

$$L_{\preceq} = \left\{ \begin{array}{|c|} \hline s \\ \hline t \\ \hline \end{array} \,\middle\|\, |s|_{\mathbf{m}} \leq |t|_{\mathbf{m}} \right\} \subseteq (\{\mathbf{m}, \mathbf{c}\} \times \{\mathbf{m}, \mathbf{c}\})^*$$

is **not regular**

Muddy Children

"After **arbitrary** but **finitely** many rounds, all the muddy children know they're muddy."

$$\varphi = \langle \exists i : \mathbf{m}_i ! \rangle \langle \underbrace{\exists i : \mathbf{m}_i \wedge \neg K_i \mathbf{m}_i}_{\psi} ! \rangle^* \bot$$

$$L_{\preceq} = \left\{ \begin{array}{|c|} \hline \mathsf{s} \\ \hline \mathsf{t} \\ \hline \end{array} \middle| \, |s|_{\mathbf{m}} \leq |t|_{\mathbf{m}} \right\} \subseteq (\{\mathbf{m}, \mathbf{c}\} \times \{\mathbf{m}, \mathbf{c}\})^*$$

is **not regular**

Counter measure: restrict to states $s \in \{\mathbf{m}\}^* \cdot \{\mathbf{c}\}^*$.

Soundness: the model and the formula are stable by permutation.

Muddy Children

"After **arbitrary** but **finitely** many rounds, all the muddy children know they're muddy."

$$\varphi = \boxed{\langle \forall i, \mathbf{m}_{i+1} \to \mathbf{m}_i ! \rangle} \langle \exists i : \mathbf{m}_i ! \rangle \langle \underbrace{\exists i : \mathbf{m}_i \wedge \neg K_i \mathbf{m}_i}_{\psi} ! \rangle^* \bot$$

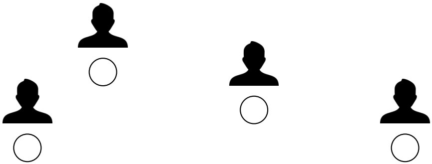
$$L_{\preceq} = \left\{ \begin{array}{|c|} \hline s \\ \hline t \\ \hline \end{array} \middle| |s|_{\mathbf{m}} \leq |t|_{\mathbf{m}} \wedge s, t \in \{\mathbf{m}\}^* \cdot \{\mathbf{c}\}^* \right\} \subseteq (\{\mathbf{m}, \mathbf{c}\} \times \{\mathbf{m}, \mathbf{c}\})^*$$

is **regular**

Counter measure: restrict to states $s \in \{\mathbf{m}\}^* \cdot \{\mathbf{c}\}^*$.

Soundness: the model and the formula are stable by permutation.

Dining Cryptographer algorithm

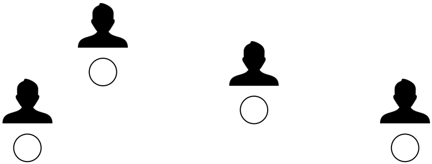


- Every cryptographer i has a private boolean p_i.
- Goal: Decide whether $\sum_i p_i > 0$ without disclosing the p_i's

Algorithm:

- For each i, sample a boolean c_i **shared** between i and $i + 1\%N$.
- Publicly announce the result of $c_i \oplus c_{i-1\%N} \oplus p_i$
- Compute $\bigoplus_i c_i$.

Dining Cryptographer algorithm



- Every cryptographer i has a private boolean p_i.
- Goal: Decide whether $\sum_i p_i > 0$ without disclosing the p_i's

Algorithm:

- For each i, sample a boolean c_i **shared** between i and $i + 1\%N$.
- Publicly announce the result of $c_i \oplus c_{i-1\%N} \oplus p_i$
- Compute $\bigoplus_i c_i$.

Simplifications: non-probabilistic setting, sequential announcements.

Mechanization

How to **mechanize** these announcements, to verify the following properties?

Formalization

Private Variables: $(p_i)_{i \in [1;N]} \in \{0,1\}^N$
Goal1: "Everyone knows whether someone paid"

$$\forall i, (K_i \exists j : p_j) \vee (K_i \forall j : \neg p_j)$$

Goal2: "No knows who paid"

$$\forall i \neq j, \neg(K_i p_j)$$

Mechanization

How to **mechanize** these announcements, to verify the following properties?

Formalization

Private Variables: $(p_i)_{i \in [1;N]} \in \{0, 1\}^N$
Goal1: "Everyone knows whether someone paid"

$$\forall i, (K_i \exists j : p_j) \vee (K_i \forall j : \neg p_j)$$

Goal2: "No knows who paid"

$$\forall i \neq j, \neg(K_i p_j)$$

"sampling random variables": $(c_i)_{i \in [1;N]}$

Mechanization

How to **mechanize** these announcements, to verify the following properties?

Formalization

Private Variables: $(p_i)_{i \in [1;N]} \in \{0,1\}^N$
Goal1: "Everyone knows whether someone paid"

$$\forall i, (K_i \exists j : p_j) \vee (K_i \forall j : \neg p_j)$$

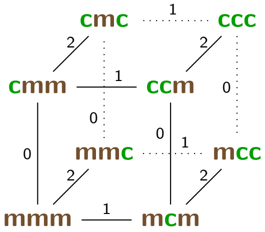
Goal2: "No knows who paid"

$$\forall i \neq j, \neg(K_i p_j)$$

"sampling random variables": $(c_i)_{i \in [1;N]}$
Announcement of agent i: **result** of computation
$r_i = c_i \oplus c_{i+1\%N} \oplus p_i$
Agents compute $\bigoplus_i r_i = \bigoplus_i p_i$

Public Announcement Whether

We introduce the new construct $\langle\varphi!!\rangle\psi$.
"After announcing **whether** there is at least one muddy child, φ"

$[\![\langle\ \exists i : \mathbf{m}_i\ !!\rangle\varphi]\!]$ (

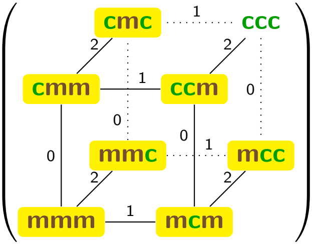
) $=$

Public Announcement Whether

We introduce the new construct $\langle\varphi!!\rangle\psi$.

"After announcing **whether** there is at least one muddy child, φ"

$[\![\langle \exists i : \mathbf{m}_i \,!!\rangle\varphi]\!]$

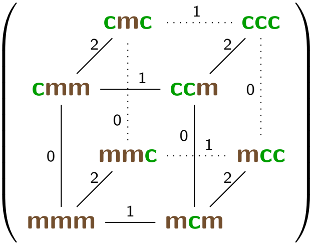


$=$

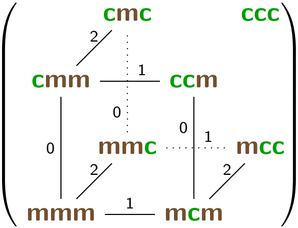
Public Announcement Whether

We introduce the new construct $\langle\varphi!!\rangle\psi$.
"After announcing **whether** there is at least one muddy child, φ"

$[\![\langle\ \exists i : \mathbf{m}_i\ !!\rangle\varphi]\!]$



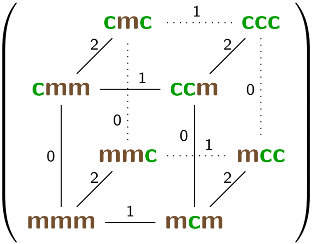
$= [\![\varphi]\!]$



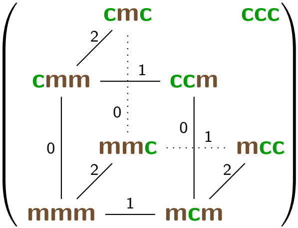
Public Announcement Whether

We introduce the new construct $\langle \varphi !! \rangle \psi$.
"After announcing **whether** there is at least one muddy child, φ"

$[\![\langle \exists i : \mathbf{m}_i !! \rangle \varphi]\!]$



$= [\![\varphi]\!]$



Good news

- Still regular: $[\![\langle \varphi !! \rangle \psi]\!](\mathcal{M}) = [\![\psi]\!](\ldots)$
- $\langle \varphi !! \rangle^* \psi$ can be computed with a disappearance relation **on pair of states**:

$$L_{\preceq} \subseteq (\Sigma' \times \Sigma')^* \quad \text{where } \Sigma' = \Sigma \times \{0,1\} \times \Sigma$$

Sequentialization

In the dining cryptographer, all announcements are different formula...

Sequentialization

In the dining cryptographer, all announcements are different formula...
or one single formula $\varphi(i)$:arameterized by $i \in Agt$.

Theorem (Addressed in Felix Thoma's BA thesis)

There exists φ' without free variables, such that

$$\langle c_0 \oplus c_1 \oplus p_0!!\rangle\langle c_1 \oplus c_2 \oplus p_1!!\rangle \dots \langle c_N \oplus c_0 \oplus p_0!!\rangle\varphi_{correct} \equiv \langle\varphi'!!\rangle^*\varphi_{correct}$$

Key ideas:

- Track the announcements already been made, by *evaluating the current common knowledge*.
- The same announcement should be made from all the states, **at the same time**.
- Solution 1: common knowledge operator $\sim^*$ (similar to safety analysis).

Sequentialization

In the dining cryptographer, all announcements are different formula...
or one single formula $\varphi(i)$:arameterized by $i \in Agt$.

Theorem (Addressed in Felix Thoma's BA thesis)

There exists φ' without free variables, such that

$$\langle c_0 \oplus c_1 \oplus p_0!!\rangle\langle c_1 \oplus c_2 \oplus p_1!!\rangle \ldots \langle c_N \oplus c_0 \oplus p_0!!\rangle\varphi_{correct} \equiv \langle\varphi'!!\rangle^*\varphi_{correct}$$

Key ideas:

- Track the announcements already been made, by *evaluating the current common knowledge*.
- The same announcement should be made from all the states, **at the same time**.
- Solution 1: common knowledge operator $\sim^*$ (similar to safety analysis).
- Solution 2: introducing $All(\varphi)$ operator, whose semantics is regular.

Summary

Regular Model Checking approach to

Knowledge Reasoning over Parameterized Systems

Summary

Learning of $\preceq$'s **regular** encoding

Regular Model Checking approach to

Knowledge Reasoning over Parameterized Systems

PPAL

$(\ )^*$ **Iterated** Announcements

Regular Models

Summary

Learning of $\preceq$'s **regular** encoding

Regular Model Checking approach to

Knowledge Reasoning over Parameterized Systems

 **PPAL**

$\big($$\big)^*$ **Iterated** Announcements

Regular Models

Future work:

- **Dynamic** Epistemic Logic (more systematic way to *dining cryptographer*, stochastic behaviours).
- **Planning**: how to synthesize announcements (card protocols).
- **Mechanize** the symmetry reductions (*Parikh images*).
- More succint models (expressing fixed point in *MONA*).

Thanks for your attention