Regular Model Checking of Epistemic Logic

Daniel Stan¹

June 27, 2022

joint work with Anthony W. Lin¹,², Felix Thoma¹
Muddy Children Puzzle (Littlewood, 1953)
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.
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 $\varphi$, 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 $M$ and some specification $\varphi$, decide whether $M \vDash \varphi$;

- **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 $\varphi$, decide whether $\mathcal{M} \vDash \varphi$;
- 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

\( m \sim c \quad m \sim c \quad (S5): \) For every \( i \), \( i \sim i \) is an equivalence relation.
Indistinguishability Relation

\[ m \sim c \]

\[(S5): \text{For every } i, i \sim \text{is an equivalence relation.}\]
Indistinguishability Relation

2 might think:
Indistinguishability Relation

2 might think:

\[ \sim \]
Indistinguishability Relation

2 might think:

(S5): For every $i$, $\sim^i$ is an equivalence relation.
Case with 3 children

From a given state $s$, from

Any agent knows the structure of the graph: common knowledge.
Kripke Structures

Case with 3 children

From a given state $s$,

- If $s \sim t$, $i$ may think we are in $t$.

Any agent knows the structure of the graph: common knowledge.
Kripke Structures

Case with 3 children

From \textit{mmm}, agent 1 knows that third letter is \textit{m}.

From a given state \( s \),

- If \( s \sim t \), \textit{i may think} we are in \( t \).
- If for all \( t \) such that \( s \sim t \), \( t \) satisfies some property \( \varphi \), then \textit{i knows} that \( \varphi \) holds (from \( s \)).
Kripke Structures

Case with 3 children

From a given state $s$,

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

\[
\begin{array}{c|c|c}
  \text{cm} & 1 & \text{cc} \\
  \hline
  0 & & 0 \\
  \text{mm} & 1 & \text{mc}
\end{array}
\]
Parameterized Verification of a property $\varphi$

$\begin{array}{c}
\text{cm} & 1 & \text{cc} \\
0 & 0 & 0 \\
\text{mm} & 1 & \text{mc}
\end{array}$

$\models \varphi$?
Parameterized Verification of a property \( \varphi \)

\[ \begin{align*}
\text{cm} & \quad 1 \quad \text{cc} \\
0 & \quad 0 \\
\text{mm} & \quad 1 \quad \text{mc} \\
\end{align*} \]

\[ \begin{align*}
\text{cmm} & \quad 2 \quad \text{cmc} \quad 1 \quad \text{ccc} \\
0 & \quad 0 \quad 0 \quad 1 \\
\text{mmm} & \quad 2 \quad \text{mcm} \\
\end{align*} \]

\[ \begin{align*}
\text{cmcm} & \quad \cdots \quad \text{ccc} \\
\text{mmcm} & \quad \cdots \quad \text{mcc} \\
\text{mmmcm} & \quad \cdots \quad \text{mmmc} \\
\text{mmmmcm} & \quad \cdots \quad \text{mcmcm} \\
\text{mmmmmcm} & \quad \cdots \quad \text{mcmcmcm} \\
\cdots & \\
\end{align*} \]
Parameterized Verification of a property $\varphi$

$\models \varphi \; \text{(for all states of all instances)}$

$\cong \text{Infinite collection of systems}$
Parameterized Verification of a property $\varphi$

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

$\leftrightarrow$ Infinite collection of systems
PAL: Public Announcement Logic (Plaza, 07)

\[ \varphi ::= p \mid T \mid \varphi \land \varphi \mid \neg \varphi \mid K_a \varphi \mid \langle \varphi! \rangle \psi \]

Where:

- \( p \in AP \) is an atomic proposition;
- \( a \in \mathbb{N} \) is a constant;
After announcing $\varphi$, $\psi$ holds

\[ a \text{ knows } \varphi \]

\[ \varphi ::= p \mid T \mid \varphi \land \varphi \mid \neg \varphi \mid K_a \varphi \mid \langle \varphi! \rangle \psi \]

Where:

$p \in AP$ is an atomic proposition;

$a \in \mathbb{N}$ is a constant;
PPAL: Parameterized Public Announcement Logic

After announcing $\varphi$, $\psi$ holds

$i$ knows $\varphi$

$\varphi ::= p_i \mid T \mid \varphi \land \varphi \mid \neg \varphi \mid K_i \varphi \mid \langle \varphi! \rangle \psi \mid \exists i : \varphi \mid i = 0 \mid i \% k = 0 \mid i = j + a$

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 $\varphi$, $\psi$ holds

- $i$ knows $\varphi$

$$\varphi ::= p_i \mid T \mid \varphi \land \varphi \mid \neg \varphi \mid K_i \varphi \mid \langle \varphi! \rangle \psi \mid \exists i : \varphi \mid i = 0 \mid i \% k = 0 \mid i = j + a$$

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 $\varphi$

$$\begin{align*}
\llbracket \varphi \rrbracket &= \begin{pmatrix}
\text{cmc} & 1 & \text{ccc} \\
2 & 0 & 2 \\
0 & 0 & 1 \\
2 & 1 & 2 \\
\text{mmm} & \text{mcm} & \text{mcc}
\end{pmatrix} \\
&= \{ s \mid s \models \varphi \}
\end{align*}$$
Semantics of a PPAL formula $\varphi$

“All children are clean”

$$\equiv \forall j: m_j \rightarrow K_j m_j$$

$$J \langle \exists i: m_i \neq \rangle \varphi \equiv \{ s \mid s \models \varphi \}$$
Semantics of a PPAL formula $\varphi$

“All children are clean”

$$\forall i : \neg m_i$$
Semantics of a PPAL formula $\varphi$

“All children are clean”

$[\forall i : \neg m_i] \left[ \begin{array}{c}
\text{cmm} \\
\text{ccc}
\end{array} \right] = \left\{ \text{ccc} \right\}$
Semantics of a PPAL formula $\varphi$

“After announcing there is at least one muddy child, all the muddy children know they’re muddy”

$\varphi \equiv \forall j : m_j \rightarrow K_j m_j$

\[
\begin{pmatrix}
\text{cmm} & \text{ccm} & \text{ccc} \\
\text{cmc} & 1 & \text{ccc} \\
\text{cm} & 2 & 0 \\
\text{mcm} & 0 & 0 \\
\text{mcc} & 0 & 0 \\
\text{mmm} & 1 & \text{mcm} \\
\end{pmatrix}
\]

$\langle \exists i : m_i ! \rangle \varphi$
Semantics of a PPAL formula $\varphi$

“After announcing there is at least one muddy child, all the muddy children know they’re muddy”

$\varphi \equiv \forall j: m_j \rightarrow K_j m_j$

$[[\exists i : m_i]!] \varphi]$ 

$[[\varphi]] = \{s | s \models \varphi\}$

$\begin{pmatrix}
\text{cmm} & \text{ccm} \\
\text{mcm} & \text{ccc} \\
\text{mmm} & \text{mcm}
\end{pmatrix} = \begin{pmatrix}
\text{cmc} & \text{ccc} \\
\text{cmc} & \text{ccc} \\
\text{mmm} & \text{mcm}
\end{pmatrix}$
Semantics of a PPAL formula $\varphi$

"After announcing there is at least one muddy child, all the muddy children know they’re muddy”

$\varphi \equiv \forall j : m_j \rightarrow K_j m_j$

$$\begin{bmatrix} 
\langle \exists i : m_i \! \rangle \varphi \rangle \\
\begin{array}{c}
cmc \\
cmm \\
mcc \\
mmm
\end{array}
\end{bmatrix} = \begin{bmatrix} 
\begin{array}{c}
cmc \\
cmm \\
mcc \\
mcc \\
mmm
\end{array}
\end{bmatrix} = \{cmc, mcc, ccm\}$$
Semantics of $\varphi$ on a paramaterized system

$$\left[ \langle \exists i : m_i! \rangle \forall j : m_j \rightarrow K_j m_j \right]$$

$$\begin{pmatrix}
\begin{array}{ccc}
\times & 1 & cc \\
0 & 0 & \\
mm & 1 & mc
\end{array}
\end{pmatrix}$$

$$\begin{pmatrix}
\begin{array}{ccc}
\times & 1 & ccc \\
2 & 0 & 1 \\
mmm & 1 & mcm
\end{array}
\end{pmatrix}$$

$$= \left\{ \begin{array}{c}
mc \\
cm \\
mcc \\
cmc \\
ccm \\
mccc \\
ccmc \\
cccm \\
\ldots
\end{array} \right\}$$
Semantics of $\varphi$ on a parameterized system

$$[[\exists i : m_i! \forall j : m_j \rightarrow K_j \cdot m_j]] = \left\{ \begin{array}{c} mc, \\
mc, \\
cc, \\
ccc, \\
cccc, \\
ccccc, \\
\ldots \end{array} \right\}$$
Semantics of $\varphi$ on a paramaterized system

$[[\exists i : m_i! \forall j : m_j \rightarrow K_j m_j]] = \{c\}^* \cdot \{m\} \cdot \{c\}^*$
Regular Encoding

\[
\begin{pmatrix}
    m_0 & m_1 \\
    c_0 & c_1 \\
\end{pmatrix}
\begin{pmatrix}
    m_0 & m_1 \\
    c_0 & c_1 \\
\end{pmatrix}
\]
Regular Encoding

Encoded as

\[
\begin{array}{cccc}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c \\
\end{array}
\]
Regular Encoding

Encoded as

\[
\begin{pmatrix}
\{m, c\} \\
\{0, 1\} \\
\{m, c\}
\end{pmatrix}^* \in \begin{pmatrix}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c
\end{pmatrix}
\]
Regular Encoding

\[
\begin{pmatrix}
0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1
\end{pmatrix}
\in
\left(\{m, c\}^* \times \{0, 1\} \times \{m, c\}\right)
\]
Regular Encoding

Encoded as

$$\begin{pmatrix}
\{m, c\}^* \\
\times \\
\{0, 1\} \\
\times \\
\{m, c\}
\end{pmatrix}$$
Regular Encoding

Encoded as

$$\begin{pmatrix}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c \\
\end{pmatrix} \in \left( \{m, c\} \right)^* \times \{0, 1\} \times \{m, c\}$$
Regular Encoding

Encoded as

\[
\begin{pmatrix}
\{m, c\} \\
\{0, 1\} \\
\{m, c\}
\end{pmatrix}^* \times 
\begin{pmatrix}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c
\end{pmatrix}
\]
Regular Encoding

Encoded as

$$\begin{pmatrix}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c
\end{pmatrix} \in \left( \left\{ m, c \right\}^* \times \left\{ 0, 1 \right\} \times \left\{ m, c \right\} \right)$$
Regular Encoding

Encoded as

\[
\begin{pmatrix}
\{m, c\} \\
\times
\{0, 1\} \\
\times
\{m, c\}
\end{pmatrix}^*
\]
Regular Encoding

Encoded as

\[
\begin{bmatrix}
\{m, c\} \\
\{0, 1\} \\
\{m, c\}
\end{bmatrix}^* 
\begin{bmatrix}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c
\end{bmatrix} 
\]
Regular Encoding

Encoded as

\[
\begin{pmatrix}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c
\end{pmatrix} 
\in 
\left(\{m, c\}\right)^* 
\times 
\{0, 1\} 
\times 
\{m, c\}
\]
Regular Encoding

Encoded as

\[
\begin{pmatrix}
m & c & m & c \\
0 & 0 & 1 & 0 \\
m & c & c & c
\end{pmatrix}
\in \left( \{m, c\} \right)^* \times \{0, 1\} \times \{m, c\}
\]
Regular Encoding

\[
\begin{array}{c}
\begin{array}{ccc}
\text{m} & \text{c} & \text{m} \\
\text{c} & \text{m} & \text{c}
\end{array}
\end{array}
\Rightarrow
\begin{array}{c}
\begin{array}{ccc}
\text{c} & \text{m} & \text{c} \\
\text{m} & \text{c} & \text{c}
\end{array}
\end{array}
\]

Encoded as

\[
\begin{pmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\in
\left(\{\text{m, c}\}\right)^* \\
\times \\
\{0, 1\} \\
\times \\
\{\text{m, c}\}
\]

\[
\begin{pmatrix}
\text{c} \\
\text{m}
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
\text{m} & \text{m} & \text{c} & \text{c} \\
1 & 1 & 1 & 1
\end{pmatrix}
\]

\[
q_0 \quad \xrightarrow{(\text{c})} \quad \begin{pmatrix}
\text{c} \\
\text{m}
\end{pmatrix}
\]

\[
q_0 \quad \xrightarrow{(\text{m})} \quad \begin{pmatrix}
\text{m} \\
\text{c}
\end{pmatrix}
\]

\[
q_0 \quad \xrightarrow{(\text{0})} \quad \begin{pmatrix}
\text{0} \\
\text{m}
\end{pmatrix}
\]

\[
q_0 \quad \xrightarrow{(\text{c})} \quad \begin{pmatrix}
\text{m} \\
\text{c}
\end{pmatrix}
\]

\[
q_1 \quad \xrightarrow{(\text{m})} \quad \begin{pmatrix}
\text{m} \\
\text{c}
\end{pmatrix}
\]

\[
q_1 \quad \xrightarrow{(\text{c})} \quad \begin{pmatrix}
\text{c} \\
\text{m}
\end{pmatrix}
\]

\[
q_1 \quad \xrightarrow{(\text{0})} \quad \begin{pmatrix}
\text{0} \\
\text{m}
\end{pmatrix}
\]

\[
q_1 \quad \xrightarrow{(\text{c})} \quad \begin{pmatrix}
\text{0} \\
\text{c}
\end{pmatrix}
\]
Definition (Regular Kripke structure)

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

- \( \Sigma \) 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 \textbf{length-preserving} transducer:

\[
T_{\mathcal{M}} = \left\{ s \otimes \left( \begin{array}{c}
0 \ldots 0 \cdot 1 \cdot 0 \ldots 0 \\
i
\end{array} \right) \otimes t \mid s \sim t \right\}
\]
Contribution: Regular Semantics of a PPAL formula

\[
\begin{array}{c|c}
0^{i-1}10^{n-i-1} & w_1 \sim w_2 \\
\hline
w_1 & w_2
\end{array}
\]

\[\{w_1 \sim w_2\} \xrightarrow{[\varphi]} \{w \mid M, w \models \varphi\}\]
Contribution: Regular Semantics of a PPAL formula

\[
\begin{align*}
&\left\{ 
\begin{array}{c}
w_1 \\
0^{i-1}10^{n-i-1} \\
w_2 
\end{array}
\right\} \\
\times
\rightarrow
\{ w \mid \mathcal{M}, w \vDash \varphi \}
\end{align*}
\]

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

\[
\begin{align*}
x + \frac{w_1}{0^{i-1}10^{n-i-1}} w_2 \\
\end{align*}
\]

\[
\begin{align*}
\left\{ \begin{array}{c}
x \\
+ \\
w_1 \\
0^{i-1}10^{n-i-1} \\
w_2 \\
\end{array} \right\} & \xrightarrow{[\phi]} \\
\left\{ \begin{array}{c}
x \\
+ \\
w \\
\end{array} \right\}
\end{align*}
\]

**Theorem**

If $M$ is a regular Kripke structure, then $[\phi](M)$ is a regular language. Moreover, the transformation is uniformly effective.

PPAL model checking is decidable.
**Contribution:** Regular Semantics of a PPAL formula

\[
\begin{align*}
\{ \begin{array}{c}
\times \\
+ \\
\hline
w_1 \\
0^{i-1}10^{n-i-1} \\
\hline
w_2
\end{array} \} \quad & \quad w_1 \sim_x w_2 \quad \longrightarrow \quad [\varphi] \\
\{ \begin{array}{c}
\times \\
+ \\
\hline
\hline
w
\end{array} \} \quad & \quad M_x, w \models \varphi
\end{align*}
\]

**Theorem**

If \( M \) is a regular Kripke structure, then \( [\varphi](M) \) 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).

Alice

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).

“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 . . .
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

$PPAL^* = PPAL + \text{iterated announcement}$:

$$\langle \varphi! \rangle^* \psi \equiv \exists k \in \mathbb{N} : \langle \varphi! \rangle \cdots \langle \varphi! \rangle \psi$$

$k$ times
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 $Init, Bad \in \text{Reg}(\Sigma)$.

Is the system *Safe*? That is to say:

$$\forall w \in Init, \forall w' \in \Sigma^*, w \sim^* w' \Rightarrow w' \notin Bad$$
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 $\text{Init}, \text{Bad} \in \text{Reg}(\Sigma)$.

Is the system *Safe*? That is to say:

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

In other words: **Decide** whether $(\text{Init} \otimes \text{Bad}) \cap T_{\sim^*} = \emptyset$, where $T_{\sim^*}$ is the transducer of the *transitive 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 $Init, Bad \in \text{Reg}(\Sigma)$.

Is the system *Safe*? That is to say:

$$\forall w \in Init, \forall w' \in \Sigma^*, w \sim^* w' \Rightarrow w' \not\in Bad$$

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

\( I \in \text{Reg}(\Sigma) \) such that

1. \( \text{Init} \subseteq I \);
2. \( I \cap \text{Bad} = \emptyset \);
3. \( \text{Post}(I) \subseteq I \)
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** $C$ describes all possible values of $H \in C$;

**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 $\Sigma$
- The **target** is a language $L \subseteq \Sigma^*$.

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

Two types of queries:

- **Membership queries:** “Does $w \in L$?” for some given $w \in \Sigma^*$
  
  **Answer:** YES or NO;

- **Equivalence 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 $B$:** $A \Delta B := A \setminus B \cup B \setminus A$. 
Example of a Learner: learnlib

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

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).
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)**

\( I \in \text{Reg}(\Sigma) \) such that

1. \( \text{Init} \subseteq I \);
2. \( I \cap \text{Bad} = \emptyset \);
3. \( \text{Post}(I) \subseteq I \)
A safety teacher

A regular invariant is defined as first-order properties.

Definition (Regular Invariant)

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

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

\( I \in \text{Reg}(\Sigma) \) such that

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

Oracle for learning some target \( I \):

- **Membership**: Given \( w \), check whether \( \exists w' \in \text{Init} : w' \sim^* w \).
- **EQuivalence queries**: Check that the three above properties hold for the hypothesis.

**Consequence**: using Angluin’s L* algorithm, the learning procedure terminates, iff \( \text{Post}^*(\text{Init}) \) is regular, or contain a bad state.
A safety teacher

A regular invariant is defined as first-order properties.

**Definition (Regular Invariant)**

\( I \in \text{Reg}(\Sigma) \) such that

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

**Oracle for learning some target \( I \):**

- **Membership:** Given \( w \), check whether \( \exists w' \in \text{Init} : w' \sim^* w \).
- **EQuivalence queries:** Check that the three above properties hold for the hypothesis.

**Consequence:** using Angluin’s L* algorithm, the learning procedure terminates, iff \( \text{Post}_\sim^*(\text{Init}) \) 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
**Muddy Children Example:** $k$ muddy children require $k$ announcements to conclude on their state.

No **fixed** PPAL formula

\[ PPAL^* = PPAL + \text{iterated announcement:} \]

\[ \langle \varphi! \rangle^* \psi \equiv \exists k \in \mathbb{N} : \langle \varphi! \rangle \ldots \langle \varphi! \rangle \psi \]

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

\[ PPAL^* = PPAL + \text{iterated announcement:} \]

\[
\langle \varphi! \rangle^k \psi \equiv \exists k \in \mathbb{N} : \langle \varphi! \rangle \ldots \langle \varphi! \rangle \psi \\
\text{k times}
\]

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 $\varphi$

$s \preceq t$ if, and only if, $\forall k, s \in S_k \Rightarrow t \in S_k$

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

$S_0 = S$
Disappearance relation for $\varphi$

$s \preceq t$ if, and only if, $\forall k, s \in S_k \Rightarrow t \in S_k$

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

$S_0 = S$

$S_1$
Disappearance relation for $\varphi$

$s \preceq t$ if, and only if, $\forall k, s \in S_k \Rightarrow t \in S_k$

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

$S_0 = S \quad S_1 \quad S_2$
Disappearance relation for $\varphi$

$s \preceq t$ if, and only if, $\forall k, s \in S_k \Rightarrow t \in S_k$

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

$s \preceq t$ if, and only if, $\forall k, s \in S_k \Rightarrow t \in S_k$

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

$S_0 = S$ $S_1$ $S_2$ $S_3$ $\ldots$ $S_\infty = \bigcap_k S_k$
Abstracting away from the iteration

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

$$\exists t \notin S_\infty \land t \preceq s \land \mathcal{M} \models_{\{u \mid t \preceq u\}} s \models \psi$$
Abstracting away from the iteration

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

\[
\exists t \notin S_\infty \land t \preceq s \land \exists k \in \mathbb{N} \quad \mathcal{M} \models \{ u \mid t \preceq u \}, s \vDash \psi
\]

Consequence: if \( \mathcal{M} \) and \( L_\preceq = \left\{ \begin{array}{c|c} s \preceq t \end{array} \right\} \) are regular,

then \( \llbracket \langle \varphi! \rangle^* \psi \rrbracket (\mathcal{M}) \) is effectively regular.
Theorem

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

Given some PPAL formula $\varphi$, 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 $s, t \in S$, $s \preceq t$;
2. An equivalence oracle: given $L'$ regular, does $L' = L_{\preceq}$ and if not, provide counterexample $w \in L \setminus L_{\preceq} \cup L_{\preceq} \setminus L$. 
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

\( \forall s \in \Sigma^k \cap S \) 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. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R \land (s_2, s_3) \in R \rightarrow (s_1, s_3) \in R$
   and

3. $\forall s, t : (s, t) \in R \lor (t, s) \in R$
   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. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R \land (s_2, s_3) \in R \rightarrow (s_1, s_3) \in R$
   and

3. $\forall s, t : (s, t) \in R \lor (t, s) \in R$
   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. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R \land (s_2, s_3) \in R \rightarrow (s_1, s_3) \in R$
3. $\forall s, t : (s, t) \in R \lor (t, s) \in R$ and
4. Some “$F$” property

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

Reflexive
Transitive
Total

FO properties over $R$
Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R \models \leq$, iff:

1. $\{ \begin{array}{c} w \\ w \end{array} \mid w \in L_S \} \subseteq L_R$

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

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

4. Some “$F$” property

All quantifications are made over $\Sigma^k \cap S$ for some fixed length $k$. 
Theorem (Unique Characterization)

For $R \subseteq S \times S$, $R \models \leq$, iff:

1. $\forall s : (s, s) \in R$
   and
2. $\forall s_1, s_2, s_3 : (s_1, s_2) \in R \land (s_2, s_3) \in R \rightarrow (s_1, s_3) \in R$
   and
3. $\forall s, t : (s, t) \in R \lor (t, s) \in R$
   and
4. Some “$F$” property

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

FO properties over $R$ 

Regular Language Queries
Theorem (Unique Characterization)

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

\[
\ldots
\]

4. Some “$F$” property
Theorem (Unique Characterization)

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

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:

\[ \ldots \]

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:
(1) \( s \) really disappears;
(2) \( s \) never disappears.
Theorem (Unique Characterization)

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

\[ \ldots \]

4. \[ \forall s \left\{ \begin{array}{l}
\text{either (1) } \forall t : (s, t) \in R \rightarrow (t, s) \notin R \iff t \in F(sR^\cdot) \\
\text{or (2) } \forall t : (s, t) \in R \rightarrow (t, s) \in R \land t \in F(sR^\cdot)
\end{array} \right. \]

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 $\{(u, v) \mid v \in F(uR\cdot)\}$

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:

\[\ldots\]

4. $\text{FO Property over } R \text{ and } \{(u, v) \mid v \in F(uR\cdot)\} \rightarrow \text{Effective and Uniformly Regular}$

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.”

\[
\varphi = \langle \exists i : m_i ! \rangle \langle \exists i : m_i \land \neg K_i m_i! \rangle^* \bot
\]
Muddy Children

“After arbitrary but finitely many rounds, all the muddy children know they’re muddy.”

\[ \varphi = \langle \exists i : m_i! \rangle \langle \exists i : m_i \land \neg K_i m_i! \rangle^\ast \perp \]

\[ L_\preceq = \left\{ \begin{array}{c|c} s & t \\ \hline m & m \end{array} \right\} \subseteq (\{m, c\} \times \{m, c\})^* \]
Muddy Children

“After arbitrary but finitely many rounds, all the muddy children know they’re muddy.”

\[ \varphi = \langle \exists i : m_i! \rangle \langle \exists i : m_i \land \neg K_i m_i! \rangle^* \perp \]

\[ L_{\subseteq} = \left\{ \begin{array}{c} s \mid s \mid m \leq t \mid m \\ s \mid m \leq t \mid m \end{array} \right\} \subseteq (\{m, c\} \times \{m, c\})^* \]

is not regular
Muddy Children

“After arbitrary but finitely many rounds, all the muddy children know they’re muddy.”

\[
\varphi = \langle \exists i : m_i! \rangle \langle \exists i : m_i \land \neg K_i m_i! \rangle^* \bot
\]

\[
L_{\leq} = \left\{ \begin{array}{ccc} s & | & m \\ t & | & m \end{array} \right\} \subseteq (\{m, c\} \times \{m, c\})^*
\]

is not regular

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

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.”

\[ \varphi = \langle \forall i, m_{i+1} \rightarrow m_i! \rangle \langle \exists i : m_i! \rangle \langle \exists i : m_i \land \neg K_i m_i! \rangle^* \perp \]

\[ L_S = \left\{ \begin{array}{c|c} s \mid s_m \leq \mid t_m \land s, t \in \{m\}^* \cdot \{c\}^* \end{array} \right\} \subseteq (\{m, c\} \times \{m, c\})^* \]

is **regular**

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

**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 $\bigoplus_i c_i$. 

Simplifications: non-probabilistic setting, sequential announcements.
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”

\[
\forall i, (K_i \exists j : p_j) \lor (K_i \forall j : \neg p_j)
\]

Goal2: “No knows who paid”

\[
\forall i \neq j, \neg (K_i p_j)
\]
Mechanization

How to \textbf{mechanize} these announcements, to verify the following properties?

\textbf{Formalization}

Private Variables: \((p_i)_{i\in[1;N]} \in \{0, 1\}^N\)

Goal 1: “Everyone knows whether someone paid”

\[
\forall i, (K_i \exists j : p_j) \lor (K_i \forall j : \neg p_j)
\]

Goal 2: “No knows who paid”

\[
\forall i \neq j, \neg(K_ip_j)
\]

“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”

\[
\forall i, (K_i \exists j : p_j) \lor (K_i \forall j : \neg p_j)
\]

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, $\varphi$”

\[
\left\lceil \langle \exists i : m_i !! \rangle \varphi \right\rceil =
\begin{pmatrix}
\text{cmc} & 1 & \text{ccc} \\
2 & 1 & 2 \\
\text{cmm} & 0 & \text{ccm} \\
0 & 0 & 0 \\
\text{mmc} & 1 & \text{mcc} \\
2 & 1 & 2 \\
\text{mmm} & 1 & \text{mcm} \\
0 & 0 & 0
\end{pmatrix}
\]
Public Announcement Whether

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

$$[[\langle \exists i : m_i!! \rangle \varphi]]$$

$$\begin{pmatrix}
\text{cmm} & \text{cmc} & \text{ccc} \\
\text{mmm} & \text{ccm} & \text{mcc} \\
\text{mmm} & \text{mcm} & \text{ccc}
\end{pmatrix} =$$
Public Announcement Whether

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

“After announcing whether there is at least one muddy child, $\varphi$”

\[
[\langle \exists i : m_i !! \rangle \varphi] = [\varphi]
\]

\[
\begin{pmatrix}
\text{cmc} & 1 & \text{ccc} \\
2 & 0 & 2 \\
\text{cmm} & 1 & \text{ccm} \\
0 & 0 & 0 \\
\text{mcm} & 1 & \text{mcc} \\
2 & 2 & 2 \\
\text{mmm} & 1 & \text{mcm} \\
0 & 0 & 0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
\text{cmc} & \text{ccc} \\
2 & 2 \\
\text{cmm} & \text{ccm} \\
0 & 0 \\
\text{mcm} & \text{mcc} \\
2 & 2 \\
\text{mmm} & \text{mcm} \\
0 & 0 \\
\end{pmatrix}
\]

Good news ▶ Still regular: $\langle \varphi!! \rangle \psi(M) = \psi(M)$ ▶ $\langle \varphi!! \rangle^* \psi$ can be computed with a disappearance relation on a pair of states: $L \preceq \subseteq (\Sigma' \times \Sigma' \times \Sigma')^*$ where $\Sigma' = \Sigma \times \{0, 1\} \times \Sigma$. 
Public Announcement Whether

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

"After announcing whether there is at least one muddy child, $\phi$"

Good news

- Still regular: $\langle \langle \phi!! \rangle \psi \rangle (M) = \langle \psi \rangle (\ldots)$

- $\langle \phi!! \rangle^* \psi$ can be computed with a disappearance relation on pair of states:

$$L \subseteq (\Sigma' \times \Sigma')^* \text{ where } \Sigma' = \Sigma \times \{0, 1\} \times \Sigma$$
Sequentialization

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

In the dining cryptographer, all announcements are different formula... or one single formula $\varphi(i)$ parameterized by $i \in \text{Agt}$.

Theorem (Addressed in Felix Thoma’s BA thesis)

There exists $\varphi'$ without free variables, such that

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

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 $\varphi(i)$ parameterized by $i \in \text{Agt}$.

**Theorem (Addressed in Felix Thoma’s BA thesis)**

*There exists $\varphi'$ without free variables, such that*

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

**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 $\text{All}(\varphi)$ operator, whose semantics is regular.
Summary

Regular Model Checking approach to
Knowledge Reasoning over Parameterized Systems

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 succinct models (expressing fixed point in MONA).
Summary

Regular Model Checking approach to Knowledge Reasoning over Parameterized Systems

Learning of $\preceq$’s \textit{regular} encoding

PPAL

Iterated Announcements

Regular Models
Summary

Regular Model Checking approach to Knowledge Reasoning over Parameterized Systems

\[ \text{PPAL} \quad \left(\text{Announcement}\right)^* \quad \text{Iterated Announcements} \]

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 succinct models (expressing fixed point in *MONA*).
Thanks for your attention