Knowledge and Distributed Coordination

Yoram Moses

Technion
Outline

- Indistinguishability and knowledge
- Modeling knowledge
- The Knowledge of Preconditions principle
- Knowledge and coordination
- Applications
Each node $i$ has an initial value $v_i$

Agent 1 must print the maximal value

After receiving “$v_2 = 100$” Agent 1 has the maximum.

Can she act?
Each node $i$ has an initial value $v_i$

Agent 1 must print the maximal value

After receiving “$v_2 = 100$” Agent 1 has the maximum.

Can she act?
Indistinguishability in Computing the Maximum ($C_T M$)

- Each node $i$ has an initial value $v_i$
- Agent 1 must print the maximal value
- After receiving “$v_2 = 100$” Agent 1 has the maximum.

Can she act?
Indistinguishability in Computing the Maximum ($\text{C}\text{T}\text{M}$)

No!
Collecting all values is not necessary

Collecting all values is not sufficient if more participants are possible.
Collecting **all** values is not necessary

Collecting all values is not sufficient if more participants are possible.
Collecting all values is not necessary

Collecting all values is not sufficient if more participants are possible
Collecting all values is not necessary if more participants are possible.
Collecting all values is not necessary

Collecting all values is not sufficient…
if more participants are possible
Collecting all values is not necessary

Collecting all values is not sufficient...

if more participants are possible
What is CtM about?

Not collecting values!

A process takes the same actions at indistinguishable points

Its actions depend on its local state.
What is CtM about?

Not collecting values!

**Indistinguishability**

A process takes the same actions at indistinguishable points.

Its actions depend on its local state.
What is CtM about?

Not collecting values!

**Indistinguishability**

A process takes the same actions at indistinguishable points

Its actions depend on its local state.
What is CtM about?

Not collecting values!

**Indistinguishability**

A process takes the same actions at indistinguishable points

Its actions depend on its local state.
Agent 1 can have the same state in different runs of the protocol
Printing \( c \) is precluded by indistinguishability.
Printing $c$ is allowed iff $\text{Max} = c$ at all indistinguishable points.
Printing $c$ is allowed iff $Max = c$ at all indistinguishable points.
Printing $c$ is allowed iff agent 1 knows that $Max = c$
Knowing that $Max = c$ is necessary and sufficient for printing $c$
Knowledge in $\text{CtM}$

Knowing that $\text{Max} = c$ can depend on:

- Messages received
- The protocol
- The possible initial values
- Network topology
- Timing guarantees re: communication, synchrony, activation
- Possibility of failures, ...
Cash from the ATM:

\[ \text{Dispense}($100) \iff \text{good credit} \]
Agreement Protocols:

\[ \text{decide}_i(\nu) \implies \text{nobody decides } \nu' \neq \nu \]
Problem Specifications and Necessary Conditions

 Autonomous Cars:

\[ \text{Enter\_intersection} \implies \text{no cross-traffic} \]
Problem Specifications and Necessary Conditions

Computing the Max:

\[ \text{print}(c) \implies \max = c \]

and we have seen

\[ \text{print}(c) \implies K_1(\max = c) \]
Computing the Max:

\[
\text{print}(c) \quad \Rightarrow \quad \text{Max} = c
\]

and we have seen

\[
\text{print}(c) \quad \Rightarrow \quad K_1(\text{Max} = c)
\]
Problem Specifications and Necessary Conditions

Computing the Max:

\[ \text{print}(c) \implies \text{Max} = c \]

and we have seen

\[ \text{print}(c) \implies K_1(\text{Max} = c) \]
The **Knowledge of Preconditions** Principle (**KoP**)
The **Knowledge of Preconditions Principle** (KoP)

If performing $\alpha \Rightarrow \varphi$

Then $i$ performs $\alpha \Rightarrow K_i \varphi$

An essential connection between knowledge and action

Let’s Prove it
The Knowledge of Preconditions Principle (KoP)

If performing $\alpha \Rightarrow \varphi$

Then $i$ performs $\alpha \Rightarrow K_i \varphi$

An essential connection between knowledge and action

Let’s Prove it
The **Knowledge of Preconditions** Principle (**KoP**)

If \( \alpha \)  
Then \( i \) performs \( \alpha \)  

\[ \mathcal{K}_i \varphi \]

An essential connection between knowledge and action

Let’s Prove it
The **Knowledge of Preconditions Principle** (KoP)

If performing $\alpha \Rightarrow \varphi$

Then $i$ performs $\alpha \Rightarrow K_i \varphi$

An essential connection between knowledge and action

Let’s Prove it
A three decades old theory of knowledge is based on

- Halpern and M. [1984]
- Parikh and Ramanujam [1985]
- Chandy and Misra [1986]
- Fagin et al. [1995], *Reasoning about Knowledge*
- and earlier Kripke 1950’s, Hintikka [1962], Aumann [1976]
Basic notion: Indistinguishability
Basic notion: Indistinguishability

\( r \)

\( r' \)

\( i \) has the same state at both points
Basic notion: **Indistinguishability**

\[ r \]

\[ r' \]

\[ Max \neq 100 \]
Basic notion: Indistinguishability

$r$

$r$

$r'$

$Max \neq 100$

$\text{print}_1(100)$
Defining Knowledge in Pictures

\[ r \]
\[ r' \]
\[ r'' \]
\[ r''' \]
\[ r^4 \]
Defining Knowledge in Pictures

\[ r \]
\[ r' \]
\[ r'' \]
\[ r''' \]
\[ r^4 \]
Defining Knowledge in Pictures

\[ r, r', r'', r''', r^4 \]

\[ \varphi \]

Knowledge of Preconditions

January 7th, 2019
Defining Knowledge in Pictures

$K_i \varphi$

$\tau$

$\tau'$

$\tau''$

$\tau'''$

$\tau^4$
Defining Knowledge more formally [Fagin et al. 1995]

- A run is a sequence $r : \mathbb{N} \rightarrow G$ of global states.
- A system is a set $R$ of runs.
- Typically, $R = \{\text{runs of a protocol } P \text{ in a model } M\}$.

Assumption

Each global state $r(t)$ determines a local state $r_i(t)$ for every agent $i$.

A point $(r, t)$ refers to time $t$ in run $r$. 
A run is a sequence \( r : \mathbb{N} \rightarrow G \) of global states.

A system is a set \( R \) of runs.

Typically, \( R = \{\text{runs of a protocol } P \text{ in a model } M\} \).

Assumption

Each global state \( r(t) \) determines a local state \( r_i(t) \) for every agent \( i \).

A point \((r, t)\) refers to time \( t \) in run \( r \).
Defining Knowledge more formally

[Fagin et al. 1995]

- A run is a sequence \( r : \mathbb{N} \rightarrow \mathcal{G} \) of global states.
- A system is a set \( R \) of runs.
- Typically, \( R = \{ \text{runs of a protocol } P \text{ in a model } M \} \).

**Assumption**

Each global state \( r(t) \) determines a local state \( r_i(t) \) for every agent \( i \).

A point \( (r, t) \) refers to time \( t \) in run \( r \).
Facts are considered "true" or "false" at a point.

\((R, r, t) \models \varphi\) denotes that \(\varphi\) is true at \((r, t)\) wrt \(R\).
A Propositional Logic of Knowledge

Starting from a set $\Phi$ of primitive propositions, define $\mathcal{L}_n^K = \mathcal{L}_n^K(\Phi)$ by

$$\varphi := p \in \Phi \mid \neg \varphi \mid \varphi \land \varphi \mid K_1 \varphi \mid \cdots \mid K_n \varphi$$

Given an interpretation $\pi : \Phi \times \text{Pts}(R) \rightarrow \{\text{True, False}\}$

$$(R, r, t) \models p, \text{ for } p \in \Phi, \text{ iff } \pi(p, r, t) = \text{True}.$$  

$$(R, r, t) \models \neg \varphi \text{ iff } (R, r, t) \not\models \varphi$$

$$(R, r, t) \models \varphi \land \psi \text{ iff both } (R, r, t) \models \varphi \text{ and } (R, r, t) \models \psi.$$
Knowledge $= \text{Truth in All Possible Worlds}$

$$(R, r, t) \models K_1 \phi \iff \text{for all points } (r', t') \text{ of } R \text{ such that } r_i(t) = r'_i(t') \text{ we have } (R, r', t') \models \phi.$$ 

Comments:

The definition ignores the complexity of computing knowledge

Local information $= \text{current local state}$

$K_1 \phi$ holds if $\phi$ is guaranteed to hold in $R$ given $i$’s local state

The definition is model independent
Knowledge = Truth in All Possible Worlds

\[(R, r, t) \models K_i \varphi \quad \text{iff} \quad \text{for all points } (r', t') \text{ of } R \text{ such that } r_i(t) = r'_i(t') \text{ we have } (R, r', t') \models \varphi.\]

Comments:

The definition ignores the complexity of computing knowledge

Local information = current local state

\(K_i \varphi\) holds if \(\varphi\) is guaranteed to hold in \(R\) given \(i\)’s local state

The definition is model independent
Knowledge $=\ Truth\ in\ All\ Possible\ Worlds$

$\ (R, r, t) \models K_i \varphi \iff\ \text{for all points } (r', t') \text{ of } R \text{ such that } r_i(t) = r_i'(t') \text{ we have } (R, r', t') \models \varphi.$

Comments:

The definition ignores the complexity of computing knowledge.

Local information $=\ current\ local\ state$

$K_i \varphi$ holds if $\varphi$ is guaranteed to hold in $R$ given $i$’s local state.

The definition is model independent.
Specifications and Knowledge

Problems in Distributed Computing are presented via specifications

- A bank’s system of ATMs
- Autonomous cars
- A distributed database
- A Google data center

Specifications impose epistemic constraints on actions!
Specifications and Knowledge

Problems in Distributed Computing are presented via specifications

- A bank’s system of ATMs
- Autonomous cars
- A distributed database
- A Google data center

Specifications impose epistemic constraints on actions!
Knowledge of Preconditions

\((R, r, t) \models \text{does}_i(\alpha) \iff i \text{ performs } \alpha \text{ at time } t \text{ in } r.\)

**Theorem (\text{KoP})**

*Under minor assumptions on \(\alpha\) and \(\varphi\) in \(R:\)*

*If \(\varphi\) is a necessary condition for \(\text{does}_i(\alpha)\) in \(R\), then \(K_i \varphi\) is a necessary condition for \(\text{does}_i(\alpha)\) in \(R\).*
Deterministic Actions

\[ r \xrightarrow{\text{does}_i(\alpha)} \]

Knowledge of Preconditions
Deterministic Actions

\[ \text{does}_i(\alpha) \]
Deterministic Actions

$\tau$

$\tau'$

$\text{does}_i(\alpha)$

$\text{does}_i(\alpha)$
Deterministic Actions

**Definition**

Action $\alpha$ is **deterministic** for $i$ in $R$ if whenever $r_i(t) = r_i'(t')$:

$$(R, r, t) \models \text{does}_i(\alpha) \text{ iff } (R, r', t') \models \text{does}_i(\alpha).$$

$i$'s local state determines whether it performs $\alpha$ at points of $R$. 
Deterministic Actions

**Definition**

Action $\alpha$ is deterministic for $i$ in $R$ if whenever $r_i(t) = r'_i(t')$: 

$$(R, r, t) \models \text{does}_i(\alpha) \quad \text{iff} \quad (R, r', t') \models \text{does}_i(\alpha).$$

$i$’s local state determines whether it performs $\alpha$ at points of $R$. 
Theorem (KoP, [M. 2015])

Let \( \alpha \) be a deterministic action for \( i \) in \( R \).

If \( \varphi \) is a necessary condition for \( \text{does}_i(\alpha) \) in \( R \),
then \( K_i \varphi \) is a necessary condition for \( \text{does}_i(\alpha) \) in \( R \).
Proof of \textbf{KoP}

\[(R, r, t) \models \text{does}_i(\alpha)\]
Proof of $\mathbf{KoP}$

\[ (r, t) \approx_i (r', t') \]
Proof of KoP

\( \alpha \) is deterministic
Proof of KoP

ϕ is a necessary condition
Proof of KoP

$\varphi$ holds at all indistinguishable points
Proof of \textbf{KoP}

\[ K_i \varphi \]

so \( K_i \varphi \) holds
Proof of KoP

\[ K_i \varphi \]
\[ \text{does}_i(\alpha) \]

\[ \Rightarrow \]
\[ K_i \varphi \]

QED
KoP Applies Very Broadly

The KoP is a universal theorem for distributed systems

KoP applies to ATMs, autonomous cars, and even more generally:

- **Legal systems:**
  
  Judge Punishes $X \implies X$ committed the crime
  
  Judge Punishes $X \implies K_J(X \text{ committed the crime})$

- **Nature:**
  
  Jellyfish stings $X \implies X \neq \text{a rock}$
  
  Jellyfish stings $X \implies K_J(X \neq \text{a rock})$

- **Betting:**
  
  Don bets on Phar Lap $\implies PL$ will win
  
  Don bets on Phar Lap $\implies K_D(\text{PL will win})$
**KoP Applies Very Broadly**

The **KoP** is a universal theorem for distributed systems.

**KoP** applies to ATMs, autonomous cars, and even more generally:

- **Legal systems:**
  
  Judge Punishes $X \Rightarrow X$ committed the crime
  
  Judge Punishes $X \Rightarrow K_J(X$ committed the crime$)$

- **Nature:**
  
  Jellyfish stings $X \Rightarrow X \neq$ a rock
  
  Jellyfish stings $X \Rightarrow K_J(X \neq$ a rock$)$

- **Betting:**
  
  Don bets on Phar Lap $\Rightarrow$ PL will win
  
  Don bets on Phar Lap $\Rightarrow K_D(PL$ will win$)$
**KoP** Applies Very Broadly

The **KoP** is a universal theorem for distributed systems

**KoP** applies to ATMs, autonomous cars, and even more generally:

- **Legal systems:**
  
  Judge Punishes $X \implies X$ committed the crime
  
  Judge Punishes $X \implies K_J(X$ committed the crime$)$

- **Nature:**
  
  Jellyfish stings $X \implies X \neq$ a rock
  
  Jellyfish stings $X \implies K_J(X \neq$ a rock$)$

- **Betting:**
  
  Don bets on Phar Lap $\implies PL$ will win
  
  Don bets on Phar Lap $\implies K_D(PL$ will win$)$
KoP Applies Very Broadly

The KoP is a universal theorem for distributed systems

KoP applies to ATMs, autonomous cars, and even more generally:

- Legal systems:
  
  Judge Punishes $X$ $\Rightarrow$ $X$ committed the crime

  Judge Punishes $X$ $\Rightarrow$ $K_J(X$ committed the crime$)$

- Nature:
  
  Jellyfish stings $X$ $\Rightarrow$ $X \neq$ a rock

  Jellyfish stings $X$ $\Rightarrow$ $K_J(X \neq$ a rock$)$

- Betting:
  
  Don bets on Phar Lap $\Rightarrow$ PL will win

  Don bets on Phar Lap $\Rightarrow$ $K_D(PL$ will win$)$
**KoP** Applies Very Broadly

The **KoP** is a universal theorem for distributed systems.

**KoP** applies to ATMs, autonomous cars, and even more generally:

- **Legal systems:**
  
  Judge Punishes $X$ $\Rightarrow$ $X$ committed the crime
  
  Judge Punishes $X$ $\Rightarrow$ $K_J(X$ committed the crime$)$

- **Nature:**
  
  Jellyfish stings $X$ $\Rightarrow$ $X \neq$ a rock
  
  Jellyfish stings $X$ $\Rightarrow$ $K_J(X \neq$ a rock$)$

- **Betting:**
  
  Don bets on **Phar Lap** $\Rightarrow$ **PL** will win
  
  Don bets on **Phar Lap** $\Rightarrow$ $K_D(PL$ will win$)$
An Application: Binary Consensus

Model:

- Each process $i = 1, \ldots, n$ starts with a value $v_i \in \{0, 1\}$.
- Communication network is a complete graph
- Synchronous message passing
- At most $t < n$ crash failures
- We assume a full-information protocol
An Application: Binary Consensus

**Specification:** A consensus protocol must guarantee

- **Decision:** Every correct process decides on a value in \( \{0, 1\} \).
- **Agreement:** All correct processes decide on the same value.
- **Validity:** A decision value must be an initial value.

Validity means \( \text{decide}_i(v) \implies \exists v \) and so

\( \text{decide}_i(0) \implies K_i \exists 0 \)
An Application: Binary Consensus

Specification: A consensus protocol must guarantee

- **Decision:** Every correct process decides on a value in $\{0, 1\}$.
- **Agreement:** All correct processes decide on the same value.
- **Validity:** A decision value must be an initial value.

Validity means $\text{decide}_i(v) \Rightarrow \exists v$

and so

$\text{decide}_i(0) \Rightarrow K_i \exists 0$
An Application: Binary Consensus

**Specification:** A consensus protocol must guarantee

- **Decision:** Every correct process decides on a value in \( \{0, 1\} \).
- **Agreement:** All correct processes decide on the same value.
- **Validity:** A decision value must be an initial value.

**Validity means**

\[
\text{decide}_i(v) \Rightarrow \exists v
\]

and so

\[
\text{decide}_i(0) \Rightarrow K_i \exists 0
\]
Knowing $\exists 0$ holds iff there is a message chain from an initial value of 0 to $j$. 

$K_j \exists 0$ holds iff there is a message chain from an initial value of 0 to $j$. 

$K_n \exists 0$ represents the condition where knowledge is present at the nth level.
Knowing $\exists 0$

How can one proc know $\exists 0$ when another does not?
Knowing $\exists 0$

How can one proc know $\exists 0$ when another does not?
Knowing $\exists 0$

How can one proc know $\exists 0$ when another does not?
Knowing $\exists 0$

How can one proc know $\exists 0$ when another does not?
Knowing $\exists 0$

How can one proc know $\exists 0$ when another does not?
Knowing $\exists 0$
Knowing $\exists \theta$
Knowing $\exists 0$

Claim: If $K_j \exists 0 \land \neg K_i \exists 0$ at time $m$, then $\geq m$ crashes have occurred.
Corollary: At time $t + 1$, either everyone knows $\exists 0$ or nobody does
A Simple Consensus Protocol

**Protocol** $P_0$ (for undecided process $i$):

if \( \text{time} = t + 1 \) \& \( K_i \exists 0 \) then decide$_i(0)$

elseif \( \text{time} = t + 1 \) \& \( \neg K_i \exists 0 \) then decide$_i(1)$

Communication is according to the fip.

All decisions at time $t + 1$
A Simple Consensus Protocol

**Protocol** $P_0$ (for undecided process $i$):

- If $\text{time} = t + 1 \land K_i \exists 0$ then $\text{decide}_i(0)$
- Elseif $\text{time} = t + 1 \land \neg K_i \exists 0$ then $\text{decide}_i(1)$

Communication is according to the fip.

All decisions at time $t + 1$
Protocol $Q_0$ (for undecided process $i$):

- **if** $K_i \exists 0$ **then** $\text{decide}_i(0)$
- **elseif** $\text{time} = t + 1 \land \neg K_i \exists 0$ **then** $\text{decide}_i(1)$

All decisions by time $t + 1$
A Better Protocol

**Protocol \( Q_0 \) (for undecided process \( i \))**:

\[
\text{if } K_i \exists 0 \text{ then } \text{decide}_i(0)
\]

\[
\text{elseif } \text{time} = t + 1 \& \neg K_i \exists 0 \text{ then } \text{decide}_i(1)
\]

All decisions by time \( t + 1 \)
Performance of $P_0$ and $Q_0$
Performance of $P_0$ and $Q_0$
Performance of $P_0$ and $Q_0$
Deciding Efficiently on 1

Design Decision: \( K_j \exists 0 \iff \text{decide}_j(0) \).

When can \( \text{decide}_i(1) \) be performed?

Recall:

**Agreement:** \( \text{decide}_i(1) \Rightarrow \text{Nobody decides 0} \)

\( \text{decide}_i(1) \Rightarrow \text{“no currently active process knows } \exists 0 \text{”} \)

By KoP, \( \text{decide}_i(1) \Rightarrow K_i(\text{nobody_knows} \exists 0) \)
Deciding Efficiently on 1

Design Decision:  \[ K_j \exists 0 \iff \text{decide}_j(0). \]

When can \( \text{decide}_i(1) \) be performed?

Recall:

**Agreement:**  \( \text{decide}_i(1) \Rightarrow \) Nobody decides 0

\[ \text{decide}_i(1) \Rightarrow \text{“no currently active process knows } \exists 0\text{”} \]

By KoP,  \( \text{decide}_i(1) \Rightarrow K_i(\text{nobody_knows} \exists 0) \)
Deciding Efficiently on 1

Design Decision: \[ K_j \exists 0 \iff \text{decide}_j(0) \].

When can \text{decide}_i(1) be performed?

Recall:

**Agreement:** \text{decide}_i(1) \implies \text{Nobody decides 0}

\text{decide}_i(1) \implies \text{“no currently active process knows } \exists 0 \text{”}

By KoP, \text{decide}_i(1) \implies K_i(\text{nobody knows } \exists 0)
Deciding Efficiently on 1

Design Decision:  \[ K_j \models 0 \iff \text{decide}_j(0). \]

When can \( \text{decide}_i(1) \) be performed?

Recall:

**Agreement:**  \( \text{decide}_i(1) \implies \text{Nobody decides 0} \)

\[ \text{decide}_i(1) \implies \text{“no currently active process knows } \exists 0 \text{”} \]

By **KoP**,  \( \text{decide}_i(1) \implies K_i(\text{nobody_knows } \exists 0) \)
Unbeatable Consensus

Protocol $OPT_0$ (for undecided process $i$):

- if $K_i \exists 0$ then decide$_i(0)$
- elseif $K_i(\text{nobody\_knows} \exists 0)$ then decide$_i(1)$

My name is Sherlock Holmes.
It is my business to know
what other people don’t know.

The Adventure of the Blue Carbuncle, 1892
Unbeatable Consensus

[Castañeda, Gonczarowski & M. ’14]

Protocol $OPT_0$ (for undecided process $i$):

if $K_i \exists 0$ then decide$_i(0)$

elseif $K_i(nobody\_knows \exists 0)$ then decide$_i(1)$

My name is Sherlock Holmes.
It is my business to know what other people don’t know.

The Adventure of the Blue Carbuncle, 1892
W.r.t. \((i, m)\), nodes are seen, crashed, or hidden.
A hidden path wrt $(i, m)$
Theorem: \( \exists \) a hidden path iff \( \neg K_i (\text{nobody knows} \exists 0) \)
Implementing $OPT_0$ [Castañeda, Gonczarowski & M. ’14]

**Standard $OPT_0$** (for undecided process $i$):

```
if seen 0 then $\text{decide}_i(0)$
elseif no hidden path then $\text{decide}_i(1)$
```

**Theorem (CGM)**

- $OPT_0$ strictly dominates $Q_0$
- $OPT_0$ is unbeatable: No consensus protocol dominates it.
- $OPT_0$ is implementable using $O(t\log n)$ bits of communication per process
Implementing $OPT_0$ [Castañeda, Gonczarowski & M. ’14]

**Standard** $OPT_0$ (for undecided process $i$):

```plaintext
if seen 0 then decide_i(0)
elseif no hidden path then decide_i(1)
```

**Theorem (CGM)**

- $OPT_0$ strictly dominates $Q_0$
- $OPT_0$ is unbeatable: No consensus protocol dominates it.
- $OPT_0$ is implementable using $O(t\log n)$ bits of communication per process
Ordering Actions

Definition (Ordered Actions)

Actions \(\langle \alpha_1, \ldots, \alpha_k \rangle\) (for agents 1, \ldots, k) are ordered in \(R\) if

\[
\text{does}_j(\alpha_j) \implies \text{Did}_{j-1}(\alpha_{j-1}) \quad \text{in } R
\]

i.e., \(t_{j-1} \leq t_j\) if \(t_i\) denotes when \(\alpha_i\) occurs.
Theorem (Nested Knowledge of Preconditions)

Let \( \langle \alpha_1, \ldots, \alpha_k \rangle \) be ordered in \( R \).

If \( \text{does}_1(\alpha_1) \Rightarrow \text{occ}'d(e) \) in \( R \)

then \( \text{does}_j(\alpha_j) \Rightarrow K_jK_{j-1} \cdots K_1 \text{occ}'d(e) \) in \( R \)
Theorem (Nested Knowledge of Preconditions)

Let \( \langle \alpha_1, \ldots, \alpha_k \rangle \) be ordered in \( R \).

If \( \text{does}_1(\alpha_1) \Rightarrow \text{occ'}d(e) \) in \( R \)

then \( \text{does}_j(\alpha_j) \Rightarrow K_jK_{j-1} \cdots K_1 \text{occ'}d(e) \) in \( R \)
In “How Processes Learn” in 1985 Chandy and Misra showed

**Theorem (Knowledge Gain)**

Let $R$ be asynchronous and $t_1 > t_0$.

If

$(R, r, t_0) \models \neg K_1 \varphi$ \hspace{1cm} and \hspace{1cm}
$(R, r, t_1) \models K_2 K_1 \varphi$

```
1   ¬K1φ

2

3

4

1 2 3 4 ¬ K 1 \varphi K 2 K 1 \varphi

\[ t_0 \hspace{1cm} t_1 \]```
Relating Knowledge and Communication

In “How Processes Learn” in 1985 Chandy and Misra showed

**Theorem (Knowledge Gain)**

Let $R$ be asynchronous and $t_1 > t_0$.

If

$(R, r, t_0) \models \neg K_1 \varphi$ and $(R, r, t_1) \models K_2 K_1 \varphi$

then there must be a (Lamport) message chain in $r$ from process 1 to process 2 between times $t_0$ and $t_1$. 

![Message chain diagram]
Relating Knowledge and Communication

In “How Processes Learn” in 1985 Chandy and Misra showed

**Theorem (Knowledge Gain)**

Let $R$ be asynchronous and $t_1 > t_0$.

If

$$ (R, r, t_0) \models \neg K_1 \varphi \quad \text{and} \quad (R, r, t_1) \models K_m K_{m-1} \cdots K_1 \varphi $$

then there must be a (Lamport) message chain in $r$ from process 1 through process 2, 3, \ldots, $m$ between times $t_0$ and $t_1$. 

Relating Knowledge and Communication

In “How Processes Learn” in 1985 Chandy and Misra showed

**Theorem (Knowledge Gain)**

Let $R$ be asynchronous and $t_1 > t_0$.

If $(R, r, t_0) \models \neg K_1 \varphi$ and 
$(R, r, t_1) \models K_m K_{m-1} \cdots K_1 \varphi$

then there must be a (Lamport) message chain in $r$ from process 1 through process 2, 3, \ldots, $m$ between times $t_0$ and $t_1$.

**Corollary**

Message chains are **necessary** for ordering actions under asynchrony.
Temporal Ordering Example: The Frozen Account

Alice, Bob, Charlie and Susan are nodes in a network.

- Alice needs to cash Charlie’s cheque
- Charlie’s account is frozen
  ⇒ they must coordinate…
Temporal Ordering Example: The Frozen Account

Alice, Bob, Charlie and Susan are nodes in a network.

- Alice needs to cash Charlie’s cheque
- Charlie’s account is frozen
  ⇒ they must coordinate...
Temporal Ordering Example: The Frozen Account

Alice, Bob, Charlie and Susan are nodes in a network.

- Alice needs to cash Charlie’s cheque
- Charlie’s account is frozen
  ⇒ they must coordinate...
Temporal Ordering Example: The Frozen Account

Alice, Bob, Charlie and Susan are nodes in a network.

- Alice needs to cash Charlie’s cheque
- Charlie’s account is frozen
  ⇒ they must coordinate...
Temporal Ordering Example: The Frozen Account

Alice, Bob, Charlie and Susan are nodes in a network.

- Alice needs to cash Charlie’s cheque
- Charlie’s account is frozen
⇒ they must coordinate...
Temporal Ordering Example: The Frozen Account

Alice, Bob, Charlie and Susan are nodes in a network.

- Alice needs to cash Charlie’s cheque
- Charlie’s account is frozen
  \[\Rightarrow\] they must coordinate...

Charlie deposits a sufficient sum
Bob reactivates Charlie’s account
Alice cashes the cheque
The Clocks and Bounds Model

We assume a directed network graph and

- Global clocks

- An upper $bound_{ij}$ on transmission times per channel $i \leftrightarrow j$
  - $1 \leq bound_{ij} < \infty$
  - Delivery within the bound is guaranteed

- Lower bounds of 1 on message transmission.
The Clocks and Bounds Model

We assume a directed network graph and

- Global clocks

- An upper $\text{bound}_{ij}$ on transmission times per channel $i \mapsto j$
  - $1 \leq \text{bound}_{ij} < \infty$
  - Delivery within the bound is guaranteed

- Lower bounds of 1 on message transmission.
The Clocks and Bounds Model

We assume a directed network graph and

- Global clocks

- An upper $bound_{ij}$ on transmission times per channel $i \leftrightarrow j$
  - $1 \leq bound_{ij} < \infty$
  - Delivery within the bound is guaranteed

- Lower bounds of 1 on message transmission.
Upper bounds determine a cone of necessarily affected nodes.
Lower bounds determine a co-cone of necessarily unaffected nodes.
Bounds create 3 regions:

- Unaffected by $\theta$
- Possibly affected by $\theta$
- Affected by $\theta$
Impact of a message delivery

Causal Cones — Digital Time/Space

Unaffected by $\theta$

$\theta$

$t$

Possibly affected by $\theta$

Affected by $\theta$
A delivery extends inner and outer regions
All past uncertainty at $t' > t$ is resolved
Ex-post, all uncertainty is resolved
Event Ordering for Alice, Bob & Charlie

- With clocks, ordering seems simple...
- But Charlie’s deposit is *spontaneous*
- Information flow is then required for
  - notifying Alice and Bob of the deposit and
  - managing coordination

- Lamport message chains can be used

Do clocks help?
Event Ordering for Alice, Bob & Charlie

- With clocks, ordering seems simple...
- But Charlie’s deposit is *spontaneous*
- Information flow is then required for
  - notifying Alice and Bob of the deposit and
  - managing coordination
- Lamport message chains can be used

Do clocks help?
Event Ordering for Alice, Bob & Charlie

- With clocks, ordering seems simple...
- But Charlie’s deposit is *spontaneous*
- Information flow is then required for
  - notifying Alice and Bob of the deposit and
  - managing coordination

- Lamport message chains can be used

Do clocks help?

Charlie deposits on **Sunday**
Bob reactivates on **Monday**
Alice cashes in on **Tuesday**
Event Ordering for Alice, Bob & Charlie

- With clocks, ordering seems simple...
- But Charlie’s deposit is *spontaneous*
- Information flow is then required for
  - notifying Alice and Bob of the deposit and
  - managing coordination

- Lamport message chains can be used

Do clocks help?
With clocks, ordering seems simple…

But Charlie’s deposit is *spontaneous*

Information flow is then required for
- notifying Alice and Bob of the deposit and
- managing coordination

Lamport message chains can be used

Do clocks help?
Event Ordering for Alice, Bob & Charlie

- With clocks, ordering seems simple...
- But Charlie’s deposit is *spontaneous*
- Information flow is then required for
  - notifying Alice and Bob of the deposit and
  - managing coordination

- Lamport message chains can be used

Do clocks help?
Ordering Based on Time Bounds

Time bounds:

Charlie deposits a sufficient sum

Alice

Bob

Charlie
t
Ordering Based on Time Bounds

Time bounds:

\[ \text{bound}_{\text{Charlie, Bob}} = 10 \]

Charlie deposits a sufficient sum
Bob reactivates Charlie’s account

Alice

Charlie

Bob

\[ t \quad t + 2 \]
Ordering Based on Time Bounds

Time bounds:

\[
\text{bound}_{\text{Charlie, Bob}} = 10
\]

Alice

Charlie

Bob

\[
t \quad t + 2 \quad t + 10
\]

Charlie deposits a sufficient sum

Bob reactivates Charlie’s account
Ordering Based on Time Bounds

Time bounds:

\[ \text{bound}_{\text{Charlie}, \text{Bob}} = 10 \]

Charlie deposits a sufficient sum

Bob reactivates Charlie’s account

Alice cashes the cheque
Susan Steps In

Time bounds:

Charlie deposits a sufficient sum

Alice

Susan

Charlie

Bob

t
Susan Steps In

Time bounds:

\[ \text{bound}_{\text{Charlie, Bob}} = 10 \]

Charlie deposits a sufficient sum

Bob reactivates Charlie’s account

\[ t \quad \text{t + 3} \]
Susan Steps In

Time bounds:

\[ \text{bound}_{\text{Charlie}, \text{Bob}} = 10 \]

Alice

Susan

Charlie

Bob

\( t \)  \( t + 3 \)  \( t + 10 \)

Charlie deposits a sufficient sum

Bob reactivates Charlie’s account

Yoram Moses ( ): Knowledge and Coordination
Susan Steps In

Time bounds:

\[ \text{bound}_{\text{Charlie, Bob}} = 10 \]
\[ \text{bound}_{\text{Susan, Bob}} = 4 \]

Charlie deposits a sufficient sum

Bob reactivates Charlie’s account

Alice

Susan

Charlie

Bob

t \quad t + 3 \quad t + 8
Susan Steps In

Time bounds:

\[
\text{bound}_{\text{Charlie}, \text{Bob}} = 10 \\
\text{bound}_{\text{Susan}, \text{Bob}} = 4
\]

Charlie deposits a sufficient sum
Bob reactivates Charlie’s account
Alice cashes the cheque
The Bound Guarantee Relation \(\rightarrow\)

Let \(D_{ij}\) = shortest bound-weighted path between \(i\) and \(j\)

Definition (Bound Guarantees)

\[\langle i, t \rangle \rightarrow \langle j, t' \rangle \quad \text{iff} \quad t' \geq t + D_{ij}\]

There is enough time from \(t\) to \(t'\) to guarantee delivery from \(i\) to \(j\)
The Bound Guarantee Relation  \(\rightarrow\)

- Let \(D_{ij}\) = shortest bound-weighted path between \(i\) and \(j\)

**Definition (Bound Guarantees)**

\[\langle i, t \rangle \rightarrow \langle j, t' \rangle \text{ iff } t' \geq t + D_{ij}\]

There is enough time from \(t\) to \(t'\) to guarantee delivery from \(i\) to \(j\)
Let $D_{ij}$ = shortest *bound*-weighted path between $i$ and $j$

**Definition (Bound Guarantees)**

$\langle i, t \rangle \rightarrow \langle j, t' \rangle$ iff $t' \geq t + D_{ij}$

There is enough time from $t$ to $t'$ to guarantee delivery from $i$ to $j$
Revisiting the Frozen Account

Message chains vs. Bound guarantees
Revisiting the Frozen Account

*Charlie notifies and coordinates both responses*

![Diagram](image-url)

- Alice
- Charlie
- Bob

$\text{Alice}$

$\text{Charlie}$

$\text{Bob}$

$t$

$t'$
Revisiting the Frozen Account

Charlie notifies both, but Susan coordinates

Alice
Susan
Charlie
Bob

\( t \)
\( t' \)
Revisiting the Frozen Account

Charlie notifies Alice, who notifies and coordinates with Bob
Revisiting the Frozen Account

Charlie notifies Bob, who notifies and coordinates with Alice.
Revisiting the Frozen Account

The four patterns are instances of

\[
\langle \text{Charlie, } t \rangle \rightarrow \theta \rightarrow \langle \text{Alice, } t' \rangle \rightarrow \langle \text{Bob, } t' \rangle
\]
Knowledge Gain with Clocks

Theorem (Ben Zvi and M.)

Let $R$ be a system with clocks and bounds and let $e$ be a spontaneous event occurring at $\langle i_0, t \rangle$ in $r \in R$. If

$$(R, r, t') \models K_2 K_1 \text{occ}'d(e)$$

then the following picture must hold in $r$: 

\[ \langle i_0, t \rangle \quad \theta \quad \langle 1, t' \rangle \quad \langle 2, t' \rangle \]
Theorem (Centipede Theorem)

Let $R$ be a system with clocks and bounds, and let $e$ be a spontaneous event occurring at $\langle i_0, t \rangle$ in $r \in R$. If

$$(R, r, t') \models K_m K_{m-1} \cdots K_1 \text{occ'd}(e)$$

then there is a centipede for $\langle i_0, 1, 2, \ldots, m \rangle$ in $r[t..t']$:
Centipedes are the analogue of message chains in this model

Centipedes are necessary for ordering action in this case
Simultaneous Actions

Definition

Actions \( \alpha_1 \) and \( \alpha_2 \) are (necessarily) simultaneous in \( R \) if both

- \( \text{does}_1(\alpha_1) \implies \text{does}_2(\alpha_2) \) and
- \( \text{does}_2(\alpha_2) \implies \text{does}_1(\alpha_1) \).

Corollaries

Let \( \alpha_1 \) and \( \alpha_2 \) be simultaneous in \( R \). Then

- \( \text{does}_1(\alpha_1) \implies K_1 \text{does}_2(\alpha_2) \) by KoP
- \( \text{does}_2(\alpha_2) \implies K_1 \text{does}_2(\alpha_2) \), so
- \( \text{does}_2(\alpha_2) \implies K_2 K_1 \text{does}_2(\alpha_2) \) by KoP
- \( \text{does}_1(\alpha_1) \implies K_2 K_1 \text{does}_2(\alpha_2) \), so
- \( \text{does}_1(\alpha_1) \implies K_1 K_2 K_1 \text{does}_2(\alpha_2) \) by KoP
- ...
Simultaneous Actions

Definition

Actions $\alpha_1$ and $\alpha_2$ are (necessarily) simultaneous in $R$ if both

- $\text{does}_1(\alpha_1) \implies \text{does}_2(\alpha_2)$ and
- $\text{does}_2(\alpha_2) \implies \text{does}_1(\alpha_1)$.

Corollaries

Let $\alpha_1$ and $\alpha_2$ be simultaneous in $R$. Then

- $\text{does}_1(\alpha_1) \implies K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_2(\alpha_2) \implies K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_2(\alpha_2) \implies K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_1(\alpha_1) \implies K_2 K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_1(\alpha_1) \implies K_1 K_2 K_1 \text{does}_2(\alpha_2)$ by KoP

...
Simultaneous Actions

Definition

Actions $\alpha_1$ and $\alpha_2$ are (necessarily) simultaneous in $R$ if both
- $\text{does}_1(\alpha_1) \implies \text{does}_2(\alpha_2)$ and
- $\text{does}_2(\alpha_2) \implies \text{does}_1(\alpha_1)$.

Corollaries

Let $\alpha_1$ and $\alpha_2$ be simultaneous in $R$. Then

- $\text{does}_1(\alpha_1) \implies K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_2(\alpha_2) \implies K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_2(\alpha_2) \implies K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_1(\alpha_1) \implies K_2 K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_1(\alpha_1) \implies K_1 K_2 K_1 \text{does}_2(\alpha_2)$ by KoP

\ldots
Simultaneous Actions

Definition

Actions $\alpha_1$ and $\alpha_2$ are (necessarily) simultaneous in $R$ if both

- $\text{does}_1(\alpha_1) \Rightarrow \text{does}_2(\alpha_2)$ and
- $\text{does}_2(\alpha_2) \Rightarrow \text{does}_1(\alpha_1)$.

Corollaries

Let $\alpha_1$ and $\alpha_2$ be simultaneous in $R$. Then

- $\text{does}_1(\alpha_1) \Rightarrow K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_2(\alpha_2) \Rightarrow K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_2(\alpha_2) \Rightarrow K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_1(\alpha_1) \Rightarrow K_2 K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_1(\alpha_1) \Rightarrow K_1 K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
- ...
Simultaneous Actions

**Definition**

Actions $\alpha_1$ and $\alpha_2$ are (necessarily) **simultaneous in** $R$ if both

- $\text{does}_1(\alpha_1) \Rightarrow \text{does}_2(\alpha_2)$ and
- $\text{does}_2(\alpha_2) \Rightarrow \text{does}_1(\alpha_1)$.

**Corollaries**

Let $\alpha_1$ and $\alpha_2$ be **simultaneous in** $R$. Then

- $\text{does}_1(\alpha_1) \Rightarrow K_1\text{does}_2(\alpha_2)$ by KoP
- $\text{does}_2(\alpha_2) \Rightarrow K_1\text{does}_2(\alpha_2)$, so
- $\text{does}_2(\alpha_2) \Rightarrow K_2K_1\text{does}_2(\alpha_2)$ by KoP
- $\text{does}_1(\alpha_1) \Rightarrow K_2K_1\text{does}_2(\alpha_2)$, so
- $\text{does}_1(\alpha_1) \Rightarrow K_1K_2K_1\text{does}_2(\alpha_2)$ by KoP
- \ldots
Simultaneous Actions

Definition

Actions $\alpha_1$ and $\alpha_2$ are (necessarily) simultaneous in $R$ if both

- $\text{does}_1(\alpha_1) \implies \text{does}_2(\alpha_2)$ and
- $\text{does}_2(\alpha_2) \implies \text{does}_1(\alpha_1)$.

Corollaries

Let $\alpha_1$ and $\alpha_2$ be simultaneous in $R$. Then

- $\text{does}_1(\alpha_1) \implies \mathcal{K}_1\text{does}_2(\alpha_2)$ by KoP
- $\text{does}_2(\alpha_2) \implies \mathcal{K}_1\text{does}_2(\alpha_2)$, so
- $\text{does}_2(\alpha_2) \implies \mathcal{K}_2\mathcal{K}_1\text{does}_2(\alpha_2)$ by KoP
- $\text{does}_1(\alpha_1) \implies \mathcal{K}_2\mathcal{K}_1\text{does}_2(\alpha_2)$, so
- $\text{does}_1(\alpha_1) \implies \mathcal{K}_1\mathcal{K}_2\mathcal{K}_1\text{does}_2(\alpha_2)$ by KoP
- $\ldots$
Simultaneous Actions

Definition

Actions $\alpha_1$ and $\alpha_2$ are (necessarily) simultaneous in $R$ if both

1. $\text{does}_1(\alpha_1) \Rightarrow \text{does}_2(\alpha_2)$ and
2. $\text{does}_2(\alpha_2) \Rightarrow \text{does}_1(\alpha_1)$.

Corollaries

Let $\alpha_1$ and $\alpha_2$ be simultaneous in $R$. Then

- $\text{does}_1(\alpha_1) \Rightarrow K_1\text{does}_2(\alpha_2)$ by KoP
- $\text{does}_2(\alpha_2) \Rightarrow K_1\text{does}_2(\alpha_2)$, so
- $\text{does}_2(\alpha_2) \Rightarrow K_2K_1\text{does}_2(\alpha_2)$ by KoP
- $\text{does}_1(\alpha_1) \Rightarrow K_2K_1\text{does}_2(\alpha_2)$, so
- $\text{does}_1(\alpha_1) \Rightarrow K_1K_2K_1\text{does}_2(\alpha_2)$ by KoP

...
Simultaneous Actions

**Definition**

Actions $\alpha_1$ and $\alpha_2$ are (necessarily) simultaneous in $R$ if both

- $\text{does}_1(\alpha_1) \implies \text{does}_2(\alpha_2)$ and
- $\text{does}_2(\alpha_2) \implies \text{does}_1(\alpha_1)$.

**Corollaries**

Let $\alpha_1$ and $\alpha_2$ be simultaneous in $R$. Then

- $\text{does}_1(\alpha_1) \implies K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_2(\alpha_2) \implies K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_2(\alpha_2) \implies K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
- $\text{does}_1(\alpha_1) \implies K_2 K_1 \text{does}_2(\alpha_2)$, so
- $\text{does}_1(\alpha_1) \implies K_1 K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
- ...
Simultaneous Actions

Definition

Actions \( \alpha_1 \) and \( \alpha_2 \) are (necessarily) simultaneous in \( R \) if both

- \( \text{does}_1(\alpha_1) \implies \text{does}_2(\alpha_2) \) and
- \( \text{does}_2(\alpha_2) \implies \text{does}_1(\alpha_1) \).

Corollaries

Let \( \alpha_1 \) and \( \alpha_2 \) be simultaneous in \( R \). Then

- \( \text{does}_1(\alpha_1) \implies K_1 \text{does}_2(\alpha_2) \) by KoP
- \( \text{does}_2(\alpha_2) \implies K_1 \text{does}_2(\alpha_2), \) so
- \( \text{does}_2(\alpha_2) \implies K_2 K_1 \text{does}_2(\alpha_2) \) by KoP
- \( \text{does}_1(\alpha_1) \implies K_2 K_1 \text{does}_2(\alpha_2), \) so
- \( \text{does}_1(\alpha_1) \implies K_1 K_2 K_1 \text{does}_2(\alpha_2) \) by KoP

...
Simultaneous Actions

Definition
Actions $\alpha_1$ and $\alpha_2$ are (necessarily) simultaneous in $R$ if both
- $\text{does}_1(\alpha_1) \Rightarrow \text{does}_2(\alpha_2)$ and
- $\text{does}_2(\alpha_2) \Rightarrow \text{does}_1(\alpha_1)$.

Corollaries
Let $\alpha_1$ and $\alpha_2$ be simultaneous in $R$. Then
1. $\text{does}_1(\alpha_1) \Rightarrow K_1 \text{does}_2(\alpha_2)$ by KoP
2. $\text{does}_2(\alpha_2) \Rightarrow K_1 \text{does}_2(\alpha_2)$, so
3. $\text{does}_2(\alpha_2) \Rightarrow K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
4. $\text{does}_1(\alpha_1) \Rightarrow K_2 K_1 \text{does}_2(\alpha_2)$, so
5. $\text{does}_1(\alpha_1) \Rightarrow K_1 K_2 K_1 \text{does}_2(\alpha_2)$ by KoP
   ...

NUS Research Week (๗)  Knowledge of Preconditions January 7th, 2019 42 / 45
Simultaneity Requires Common Knowledge

The agents in $G$ have common knowledge of $\varphi$, denoted by $C_G\varphi$, if

$$K_{i_1}K_{i_2}\cdots K_{i_m}\varphi$$

holds for all sequences $\langle i_1, i_2, \ldots, i_m \rangle$ of agents in $G$, for all $m > 0$.

**Theorem (Common Knowledge of Preconditions)**

Suppose that $A = \{\alpha_i\}_{i \in G}$ are simultaneous actions in $R$.

If $\varphi$ is a necessary condition for $\text{does}_i(\alpha_i)$ for some $i \in G$, then $C_G\varphi$ is a necessary condition for $\text{does}_j(\alpha_j)$, for all $j \in G$.

cf. [Halpern and M. '90]
Simultaneity Requires Common Knowledge

The agents in \( G \) have common knowledge of \( \varphi \), denoted by \( C_G \varphi \), if

\[
K_{i_1} K_{i_2} \cdots K_{i_m} \varphi
\]

holds for all sequences \( \langle i_1, i_2, \ldots, i_m \rangle \) of agents in \( G \), for all \( m > 0 \).

**Theorem (Common Knowledge of Preconditions)**

Suppose that \( A = \{ \alpha_i \}_{i \in G} \) are simultaneous actions in \( R \).

If \( \varphi \) is a necessary condition for \( \text{does}_i(\alpha_i) \) for some \( i \in G \), then \( C_G \varphi \) is a necessary condition for \( \text{does}_j(\alpha_j) \), for all \( j \in G \).

cf. [Halpern and M. ’90]
Knowledge and Coordination

Individual Action ⇔ Knowledge of Preconditions (KoP)

Ordered Action ⇔ Nested Knowledge of Preconditions

Simultaneous Action ⇔ Common Knowledge of Preconditions
Summary

- The KoP relates knowledge and action

- Knowledge is defined in a model-independent fashion

- Applies very broadly: Social science, Life sciences, VLSI design...

- Useful for designing efficient distributed protocols

- Effective for analyzing coordination

- Next step: Probabilistic variants of the KoP.
Summary

- The KoP relates knowledge and action
- Knowledge is defined in a model-independent fashion
- Applies very broadly: Social science, Life sciences, VLSI design...
- Useful for designing efficient distributed protocols
- Effective for analyzing coordination
- Next step: Probabilistic variants of the KoP.

Thank You!
Summary

- The **KoP** relates knowledge and action
- Knowledge is defined in a model-independent fashion
- Applies very broadly: Social science, Life sciences, VLSI design...
- Useful for designing efficient distributed protocols
- Effective for analyzing coordination
- Next step: Probabilistic variants of the **KoP**.

Thank You!
Summary

- The KoP relates knowledge and action
- Knowledge is defined in a model-independent fashion
- Applies very broadly: Social science, Life sciences, VLSI design...
- Useful for designing efficient distributed protocols
- Effective for analyzing coordination
- Next step: Probabilistic variants of the KoP.

Thank You!
Summary

- The KoP relates knowledge and action
- Knowledge is defined in a model-independent fashion
- Applies very broadly: Social science, Life sciences, VLSI design...
- Useful for designing efficient distributed protocols
- Effective for analyzing coordination
- Next step: Probabilistic variants of the KoP.

Thank You!
Summary

- The KoP relates knowledge and action
- Knowledge is defined in a model-independent fashion
- Applies very broadly: Social science, Life sciences, VLSI design...
- Useful for designing efficient distributed protocols
- Effective for analyzing coordination
- Next step: Probabilistic variants of the KoP.

Thank You!
Summary

- The **KoP** relates knowledge and action
- Knowledge is defined in a model-independent fashion
- Applies very broadly: Social science, Life sciences, VLSI design...
- Useful for designing efficient distributed protocols
- Effective for analyzing coordination

**Next step**: Probabilistic variants of the **KoP**.

Thank You!
Summary

- The KoP relates knowledge and action
- Knowledge is defined in a model-independent fashion
- Applies very broadly: Social science, Life sciences, VLSI design...
- Useful for designing efficient distributed protocols
- Effective for analyzing coordination
- **Next step:** Probabilistic variants of the KoP.

Thank You!
Chandy, Mani and Jay Misra.  
**How Processes Learn.**  
Distributed Computing, 1986.

**Reasoning About Knowledge.**  

Ben-Zvi, Ido, and YM.  
**Beyond Lamport’s Happened-before: On Time Bounds and the Ordering of Events in Distributed Systems.**  

Castañeda, Armando, Yannai A. Gonczarowski, and YM.  
**Unbeatable Consensus.**  
Proceedings of DISC 2014.

Moses, Yoram.  
**Relating Knowledge and Coordinated Action: The Knowledge of Preconditions Principle.**  

Goren, Guy, and YM.  
**Silence.**  
Proceedings of PODC 2018