Knowledge and Action in DA and BDA

Yoram Moses

Technion

Many Models of Distributed Computing

DC research is typically performed in a given model.

Many models of interest:

- Synchronous: LOCAL, CONGEST, Congested CLIQUE, ...
- Asynchronous: message-passing, shared memory, Iterated Snapshot...
- Partial synchrony
- Faults: none, crash, omission, Byzantine,...
- Population protocols, Finite-state models, Amoebot, Stone-Age, ...
- Multitude of possible models for biological systems

Can we formally capture common themes?

Many Models of Distributed Computing

DC research is typically performed in a given model.

Many models of interest:

- Synchronous: LOCAL, CONGEST, Congested CLIQUE, ...
- Asynchronous: message-passing, shared memory, Iterated Snapshot...
- Partial synchrony
- Faults: none, crash, omission, Byzantine,...
- Population protocols, Finite-state models, Amoebot, Stone-Age, ...
- Multitude of possible models for biological systems

Can we formally capture common themes?

Many Models of Distributed Computing

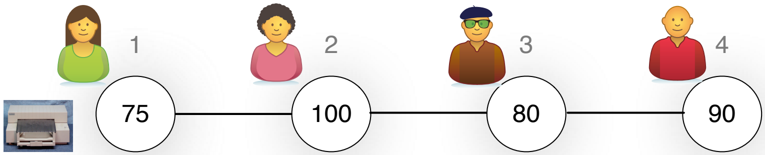
DC research is typically performed in a given model.

Many models of interest:

- Synchronous: LOCAL, CONGEST, Congested CLIQUE, ...
- Asynchronous: message-passing, shared memory, Iterated Snapshot...
- Partial synchrony
- Faults: none, crash, omission, Byzantine,...
- Population protocols, Finite-state models, Amoebot, Stone-Age, ...
- Multitude of possible models for biological systems

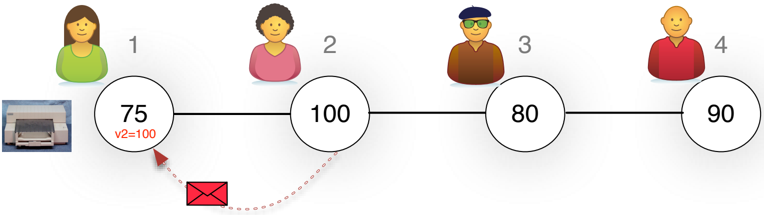
Can we formally capture common themes?

Computing the Maximum (CtM)



- 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?

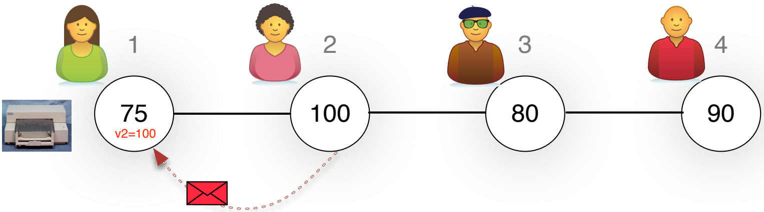
Computing the Maximum (CTM)



- 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.

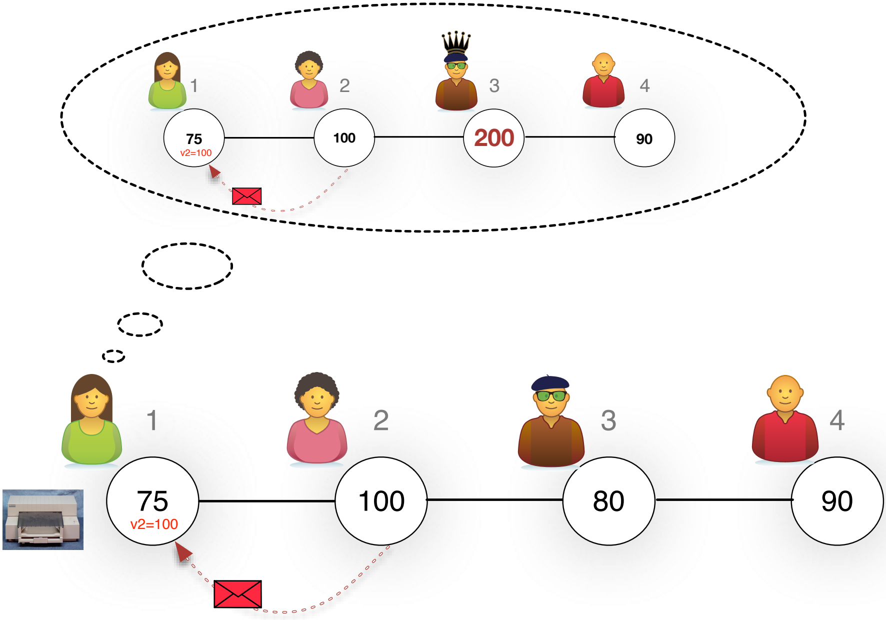
  Can she act?

Computing the Maximum (CTM)

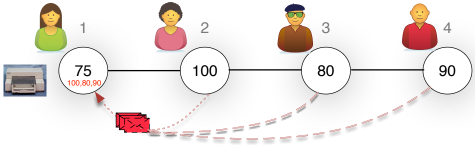


- Each node i has an initial value v_i
- Agent 1 must print the maximal value
- After receiving "$v_2 = 100$" Agent 1 has the maximum.

  Can she act?


1
2
3
4
75
v2=100
100
200
90
1
2
3
4
75
v2=100
100
80
90

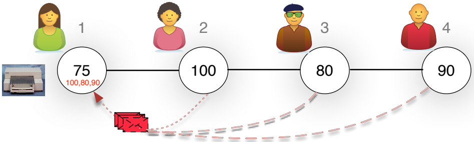
Collecting Values



Even collecting all values is not sufficient

if more participants are possible

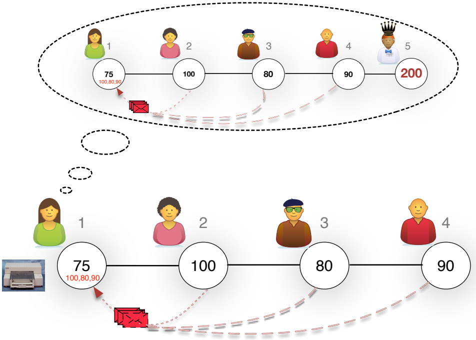
Collecting Values



Even collecting all values is not sufficient...

if more participants are possible

Collecting Values



Even collecting all values is not sufficient...

if more participants are possible

What is CTM about?

Not collecting values!

Printing c is forbidden if some indistinguishable point satisfies $Max \neq c$.

What is CTM about?

Not collecting values!

Indistinguishability and **Knowledge**

Printing c is forbidden if some indistinguishable point satisfies $Max \neq c$.

What is CTM about?

Not collecting values!

Indistinguishability and **Knowledge**

Printing c is forbidden if some indistinguishable point satisfies $Max \neq c$.

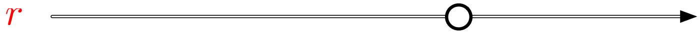
A Theory of Knowledge in Distributed Systems

A three decades old theory of knowledge is based on

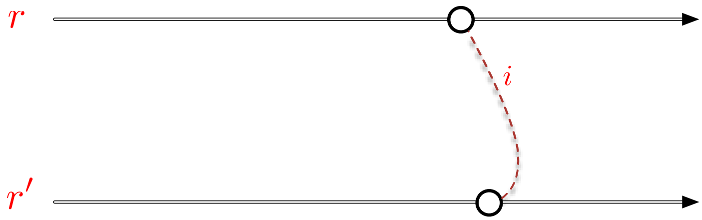
- Halpern and M. [1984]
- Parikh and Ramanujam [1985]
- Chandy and Misra [1986]

- Fagin *et al.* [1995], Reasoning about Knowledge
- and earlier Kripke 1950's, Hintikka [1962], Aumann [1976]

Basic notion: Indistinguishability

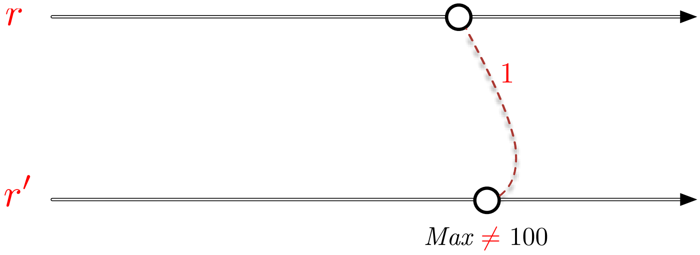


Basic notion: Indistinguishability

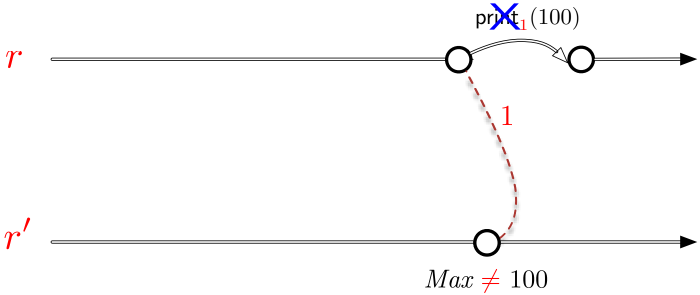


i has the same state at both points

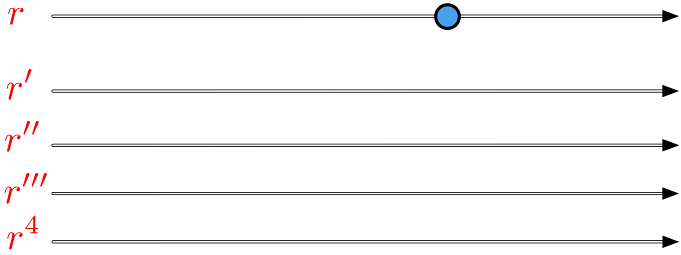
Basic notion: Indistinguishability



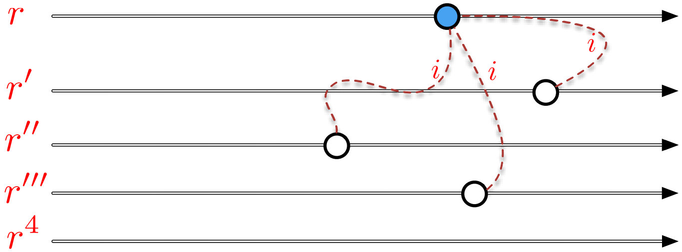
Basic notion: Indistinguishability



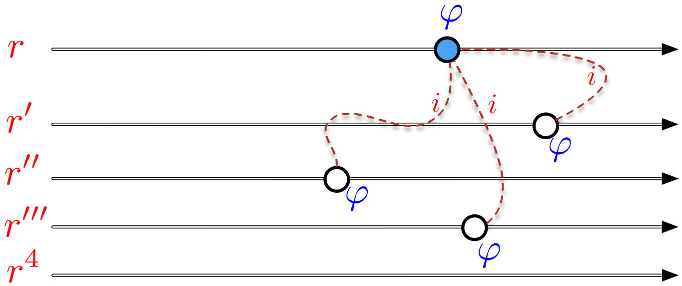
Defining Knowledge in Pictures



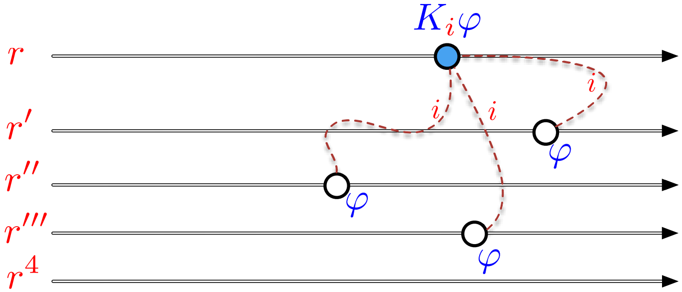
Defining Knowledge in Pictures



Defining Knowledge in Pictures



Defining Knowledge in Pictures



Defining Knowledge more formally [Fagin *et al.* 1995]

- A run is a sequence $r : \mathbb{N} \to \mathcal{G}$ of global states.
- A system is a set R of runs.

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 "true" or "false" at a point.

$(R, r, t) \vDash \varphi$ denotes that φ is true at (r, t) wrt R.

Defining Knowledge more formally [Fagin *et al.* 1995]

- A run is a sequence $r : \mathbb{N} \to \mathcal{G}$ of global states.
- A system is a set R of runs.

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 "true" or "false" at a point.

$(\boldsymbol{R}, \boldsymbol{r}, \boldsymbol{t}) \vDash \varphi$ denotes that φ is true at (r, t) wrt R.

Defining Knowledge more formally [Fagin *et al.* 1995]

- A run is a sequence $r : \mathbb{N} \to \mathcal{G}$ of global states.
- A system is a set R of runs.

  R is the set of all runs of a protocol in a given model.

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 "true" or "false" at a point

$(R, r, t) \models \varphi$ denotes that φ is true at (r, t) wrt R

Defining Knowledge more formally [Fagin *et al.* 1995]

- A run is a sequence $r : \mathbb{N} \to \mathcal{G}$ of global states.
- A system is a set R of runs.

  R is the set of all runs of a protocol in a given model.

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 "true" or "false" at a point

$(\boldsymbol{R}, \boldsymbol{r}, \boldsymbol{t}) \vDash \boldsymbol{\varphi}$ denotes that φ is true at (r, t) wrt R

Knowledge = Truth in All Possible Worlds

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

Comments:

The definition ignores the complexity of computing knowledge

Local information = current local state

The definition is model independent, and so especially interesting for BDA

Knowledge = Truth in All Possible Worlds

$$(\boldsymbol{R}, \boldsymbol{r}, \boldsymbol{t}) \vDash \boldsymbol{K_i}\varphi \quad \text{iff} \quad (R, r', t') \vDash \varphi \text{ for all points } (r', t') \text{ of } R \text{ such that } r_i(t) = r'_i(t').$$

Comments:

The definition ignores the complexity of computing knowledge

Local information = current local state

The definition is model independent, and so especially interesting for BDA

Knowledge = Truth in All Possible Worlds

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

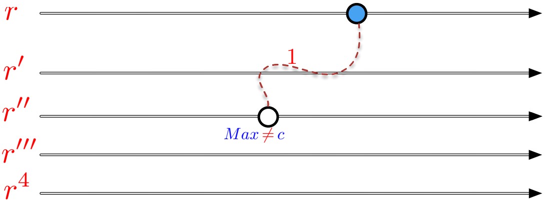
Comments:

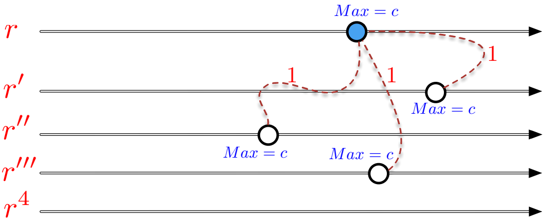
The definition ignores the complexity of computing knowledge

Local information = current local state

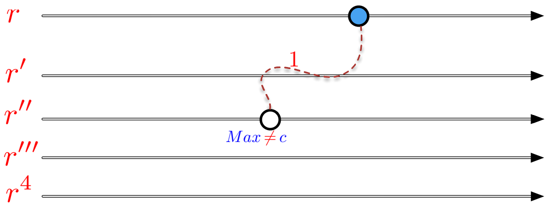
The definition is model independent, and so especially interesting for BDA

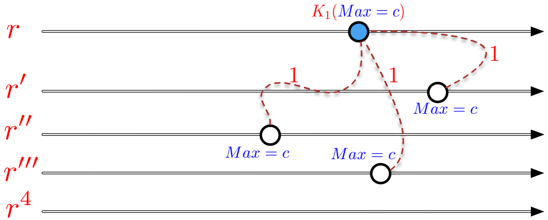
Knowledge is the Dual of Indistinguishability





Knowledge is the Dual of Indistinguishability





- **Knowing** that $Max = c$ is necessary and sufficient for printing c

- This is the single unifying property of all solutions to CTM

- **Knowing** that $Max = c$ is necessary and sufficient for printing c

- This is the single unifying property of all solutions to CTM

Knowledge in CTM

Knowing that $Max = c$ can depend on:

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

Problem Specifications and Necessary Conditions

Cash from the ATM:

$$\text{Dispense}(\$100) \quad \Rightarrow \quad \text{good credit}$$

Problem Specifications and Necessary Conditions

Agreement Protocols:

$$\text{Decide}_i(v) \quad \Rightarrow \quad \text{nobody decides } v' \neq v$$

Problem Specifications and Necessary Conditions

Autonomous Cars:

$$\text{Enter\_intersection} \quad \Rightarrow \quad \text{no cross-traffic}$$

Problem Specifications and Necessary Conditions

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:

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

and we have seen

$$\text{print}(c) \quad \Rightarrow \quad K_1(\text{Max} = c)$$

Problem Specifications and Necessary Conditions

Computing the Max:

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

and we have seen

$$\text{print}(c) \quad \Rightarrow \quad K_1(\text{Max} = c)$$

The **Knowledge of Preconditions** Principle (**K*o*P**)

If performing $\alpha \Rightarrow \varphi$

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

An essential connection between knowledge and action

The **Knowledge of Preconditions** Principle (**K*o*P**)

If	performing α	$\Rightarrow$	φ
Then	i performs α	$\Rightarrow$	$K_i\varphi$

An essential connection between knowledge and action

The **Knowledge of Preconditions** Principle (**KoP**)

If	performing α	$\Rightarrow$	φ
Then	i performs α	$\Rightarrow$	$K_i\varphi$

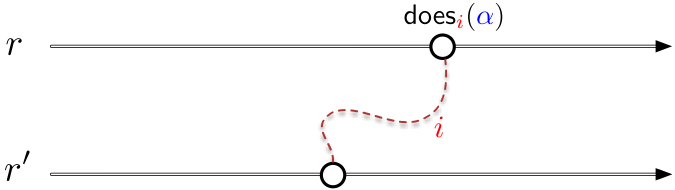
An essential connection between knowledge and action

State-enabled Actions

Definition

Action α is state-enabled in R when for all points (r, t) and (r', t') of R, if $r_i(t) = r'_i(t')$ & $(R, r, t) \models \mathsf{does}_i(\alpha)$ then there is a run r'' that agrees with r' up to time t' such that $(R, r'', t') \models \mathsf{does}_i(\alpha)$.

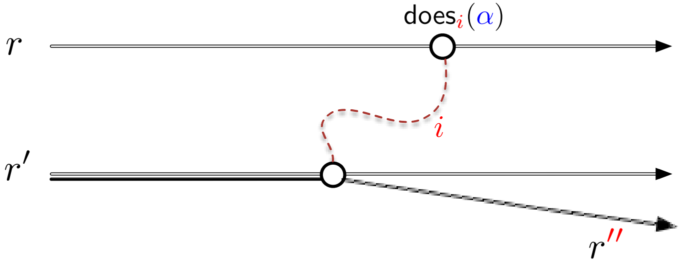
State-enabled Actions



Definition

Action $\boldsymbol{\alpha}$ is state-enabled in R when for all points (r, t) and (r', t') of R, if $r_i(t) = r'_i(t')$ & $(R, r, t) \vDash \texttt{does}_i(\alpha)$ then there is a run r'' that agrees with r' up to time t' such that $(R, r'', t') \vDash \texttt{does}_i(\alpha)$.

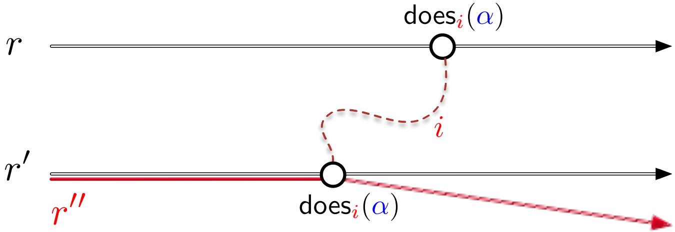
State-enabled Actions



Definition

Action α is state-enabled in R when for all points (r, t) and (r', t') of R, if $r_i(t) = r'_i(t')$ & $(R, r, t) \models \texttt{does}_i(\alpha)$ then there is a run r'' that agrees with r' up to time t' such that $(R, r'', t') \models \texttt{does}_i(\alpha)$.

State-enabled Actions



Definition

Action α is state-enabled in R when for all points (r, t) and (r', t') of R, if $r_i(t) = r'_i(t')$ & $(R, r, t) \models \mathtt{does}_i(\alpha)$ then there is a run r'' that agrees with r' up to time t' such that $(R, r'', t') \models \mathtt{does}_i(\alpha)$.

The **K**o**P** Theorem

$(R, r, t) \vDash \mathtt{does}_i(\alpha)$ iff i performs α at time t in r.

Theorem (KoP)

Assume that α is a state-enabled action in R.

If	φ	*is a necessary condition for* $\mathtt{does}_i(\alpha)$ *in* R,
then	$K_i\varphi$	*is a necessary condition for* $\mathtt{does}_i(\alpha)$ *in* R.

Actions by ATM's, Autonomous cars, Consensus, require knowing their preconditions.

The **K*o*P** Theorem

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

Theorem (K*o*P)

Assume that α is a state-enabled action in R.

If	φ	*is a necessary condition for* $\texttt{does}_i(\alpha)$ *in* R,
then	$K_i\varphi$	*is a necessary condition for* $\texttt{does}_i(\alpha)$ *in* R.

Actions by ATM's, Autonomous cars, Consensus, require knowing their preconditions.

The Scope of **K*o*P**

The **K*o*P** is a universal theorem for distributed systems

K*o*P applies even more generally:

- Legal systems:

  Judge Punishes $X \Rightarrow X$ committed the crime

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

- Betting:

  Don bets on Phar Lap $\Rightarrow$ PL will win

  Don bets on Phar Lap $\Rightarrow K_D($PL will win$)$

The Scope of **K*o*P**

The **K*o*P** is a universal theorem for distributed systems

K*o*P applies even more generally:

- Legal systems:

  Judge Punishes $X \Rightarrow X$ committed the crime

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

- Betting:

  Don bets on `Phar Lap` $\Rightarrow$ `PL` will win

  Don bets on `Phar Lap` $\Rightarrow K_D($`PL` will win$)$

The Scope of **K*o*P**

The **K*o*P** is a universal theorem for distributed systems

K*o*P applies even more generally:

- Legal systems:

$$\text{Judge Punishes } X \quad \Rightarrow \quad X \text{ committed the crime}$$

$$\text{Judge Punishes } X \quad \Rightarrow \quad K_J(X \text{ committed the crime})$$

- Betting:

$$\text{Don bets on } \texttt{Phar Lap} \quad \Rightarrow \quad \texttt{PL} \text{ will win}$$

$$\text{Don bets on } \texttt{Phar Lap} \quad \Rightarrow \quad K_D(\texttt{PL} \text{ will win})$$

The Scope of KoP

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

KoP applies even more generally:

- Legal systems:

  Judge Punishes $X \Rightarrow X$ committed the crime

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

- Betting:

  Don bets on `Phar Lap` $\Rightarrow$ `PL` will win

  Don bets on `Phar Lap` $\Rightarrow K_D($`PL` will win$)$

KoP in Biological Systems

A jellyfish behaves in a manner satisfying:

$$\text{Jellyfish stings } X \quad \Rightarrow \quad X \neq \text{a rock}$$

$$\text{Jellyfish stings } X \quad \Rightarrow \quad K_J(X \neq \text{a rock})$$

Natural organisms "implement protocols", and
their bahavior "satisfies specification"

This imposes constraints on the information that they must use

KoP in Biological Systems

A jellyfish behaves in a manner satisfying:

$$\text{Jellyfish stings } X \quad \Rightarrow \quad X \neq \text{a rock}$$

$$\text{Jellyfish stings } X \quad \Rightarrow \quad K_J(X \neq \text{a rock})$$

Natural organisms "implement protocols", and

their bahavior "satisfies specification"

This imposes constraints on the information that they must use

KoP in Biological Systems

A jellyfish behaves in a manner satisfying:

$$\text{Jellyfish stings } X \quad \Rightarrow \quad X \neq \text{a rock}$$

$$\text{Jellyfish stings } X \quad \Rightarrow \quad K_J(X \neq \text{a rock})$$

Natural organisms "implement protocols", and

their bahavior "satisfies specification"

This imposes constraints on the information that they must use

Ordering Actions

Definition (Ordered Actions)

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

$$\mathtt{does}_j(\boldsymbol{\alpha_j}) \quad \Rightarrow \quad \mathtt{Did}_{j-1}(\boldsymbol{\alpha_{j-1}}) \qquad \text{in } R$$

$t_{j-1} \leq t_j$ if each action α_i occurs at time t_i

Nested Knowledge and Ordered Actions

Theorem (Nested Knowledge of Preconditions)

Let $\langle \boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_k} \rangle$ be ordered in R.

If	$\mathtt{does}_1(\boldsymbol{\alpha_1})$	$\Rightarrow$	$\mathrm{occ'd}(e)$	in R
then	$\mathtt{does}_j(\boldsymbol{\alpha_j})$	$\Rightarrow$	$K_j K_{j-1} \cdots K_1 \ \mathrm{occ'd}(e)$	in R

Knowledge Gain

Theorem (Chandy & Misra 1986)

Let R be asynchronous and $t_1 > t_0$.

If $(R, r, t_0) \vDash \neg K_1\varphi$ *and*

$(R, r, t_1) \vDash K_2 K_1\varphi$

Knowledge Gain

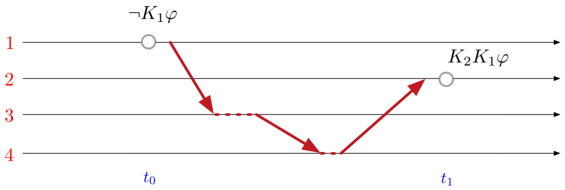
Theorem (Chandy & Misra 1986)

Let R be asynchronous and $t_1 > t_0$.

If $(R, r, t_0) \vDash \neg K_1\varphi$ *and*

$(R, r, t_1) \vDash K_2 K_1\varphi$

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



Knowledge Gain

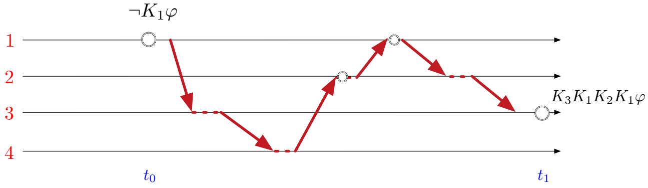
Theorem (Chandy & Misra 1986)

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, ..., m between times t_0 and t_1.



Knowledge Gain

Theorem (Chandy & Misra 1986)

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, ..., m between times t_0 and t_1.

Corollary

The only way to coordinate ordered actions under asynchrony is by constructing message chains.

Knowledge and Coordination: Simultaneous Actions

Definition

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

- $\texttt{does}_2(\boldsymbol{\alpha_2})$ is a necessary condition for $\texttt{does}_1(\boldsymbol{\alpha_1})$ and
- $\texttt{does}_1(\boldsymbol{\alpha_1})$ is a necessary condition for $\texttt{does}_2(\boldsymbol{\alpha_2})$.

Corollaries

Let α_1 and α_2 be simultaneous in R. Then

$K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$

$K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_2(\alpha_2)$, so

$K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_2(\alpha_2)$,

$K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$, so

$K_1K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$,

$K_2K_1K_2K_1\texttt{does}_2(\ \ldots$

Knowledge and Coordination: Simultaneous Actions

Definition

Actions α_1 and α_2 are (necessarily) simultaneous in R if both

- $\mathtt{does}_2(\alpha_2)$ is a necessary condition for $\mathtt{does}_1(\alpha_1)$ and
- $\mathtt{does}_1(\alpha_1)$ is a necessary condition for $\mathtt{does}_2(\alpha_2)$.

Corollaries

Let α_1 and α_2 be simultaneous in R. Then

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$, so

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$,

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$, so

$K_1K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$,

$K_2K_1K_2K_1\mathtt{does}_2(\ldots$

Knowledge and Coordination: Simultaneous Actions

Definition

Actions α_1 and α_2 are (necessarily) simultaneous in R if both

- $\texttt{does}_2(\alpha_2)$ is a necessary condition for $\texttt{does}_1(\alpha_1)$ and
- $\texttt{does}_1(\alpha_1)$ is a necessary condition for $\texttt{does}_2(\alpha_2)$.

Corollaries

Let α_1 and α_2 be simultaneous in R. Then

$K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$

$K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_2(\alpha_2)$, so

$K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_2(\alpha_2)$,

$K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$, so

$K_1K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$,

$K_2K_1K_2K_1\texttt{does}_2(\ \ldots$

Knowledge and Coordination: Simultaneous Actions

Definition

Actions α_1 and α_2 are (necessarily) simultaneous in R if both

- $\mathtt{does}_2(\alpha_2)$ is a necessary condition for $\mathtt{does}_1(\alpha_1)$ and
- $\mathtt{does}_1(\alpha_1)$ is a necessary condition for $\mathtt{does}_2(\alpha_2)$.

Corollaries

Let α_1 and α_2 be simultaneous in R. Then

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$, so

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$,

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$, so

$K_1K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$,

$K_2K_1K_2K_1\mathtt{does}_2(\ \ldots$

Knowledge and Coordination: Simultaneous Actions

Definition

Actions α_1 and α_2 are (necessarily) simultaneous in R if both

- $\mathtt{does}_2(\alpha_2)$ is a necessary condition for $\mathtt{does}_1(\alpha_1)$ and
- $\mathtt{does}_1(\alpha_1)$ is a necessary condition for $\mathtt{does}_2(\alpha_2)$.

Corollaries

Let α_1 and α_2 be simultaneous in R. Then

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$, so

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$,

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$, so

$K_1K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$,

$K_2K_1K_2K_1\mathtt{does}_2($ ...

Knowledge and Coordination: Simultaneous Actions

Definition

Actions α_1 and α_2 are (necessarily) simultaneous in R if both

- $\mathtt{does}_2(\alpha_2)$ is a necessary condition for $\mathtt{does}_1(\alpha_1)$ and
- $\mathtt{does}_1(\alpha_1)$ is a necessary condition for $\mathtt{does}_2(\alpha_2)$.

Corollaries

Let α_1 and α_2 be simultaneous in R. Then

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$

$K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$, so

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_2(\alpha_2)$,

$K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$, so

$K_1K_2K_1\mathtt{does}_2(\alpha_2)$ is a nec. condition for $\mathtt{does}_1(\alpha_1)$,

$K_2K_1K_2K_1\mathtt{does}_2(\ldots$

Knowledge and Coordination: Simultaneous Actions

Definition

Actions α_1 and α_2 are (necessarily) simultaneous in R if both

- $\texttt{does}_2(\alpha_2)$ is a necessary condition for $\texttt{does}_1(\alpha_1)$ and
- $\texttt{does}_1(\alpha_1)$ is a necessary condition for $\texttt{does}_2(\alpha_2)$.

Corollaries

Let α_1 and α_2 be simultaneous in R. Then

$K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$

$K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_2(\alpha_2)$, so

$K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_2(\alpha_2)$,

$K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$, so

$K_1K_2K_1\texttt{does}_2(\alpha_2)$ is a nec. condition for $\texttt{does}_1(\alpha_1)$,

$K_2K_1K_2K_1\texttt{does}_2($ ...

Simultaneity Requires Common Knowledge

Theorem (Common Knowledge of Preconditions)

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

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

cf. [Halpern and M. '90]

Knowledge and Coordination

Individual Action	$\Leftrightarrow$	Knowledge of Preconditions (**KoP**)
Ordered Action	$\Leftrightarrow$	Nested Knowledge of Preconditions
Simultaneous Action	$\Leftrightarrow$	Common Knowledge of Preconditions

Conclusions

There is a close connection between knowledge and coordination

K*o*P formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **K*o*P** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

KoP formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **KoP** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

KoP formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **KoP** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

KoP formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **KoP** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

KoP formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **KoP** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

K*o*P formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **K*o*P** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

K*o*P formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **K*o*P** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

K*o*P formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **K*o*P** and their applications

Much is left to be explored

Thank You!

Conclusions

There is a close connection between knowledge and coordination

K*o*P formally relates knowledge and action, applies universally

It allows fairly model-independent analysis

Diverse applications including standard distributed algorithms, VLSI, real-time coordination

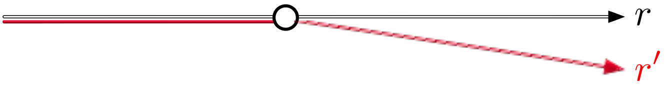
Can provide insights into the analysis of biological distributed systems

Open: Stochastic variants of **K*o*P** and their applications

Much is left to be explored

Thank You!

Past-dependent Formulas

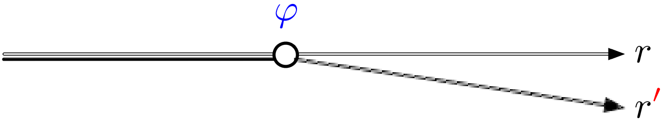


Definition

φ is past-dependent in R if for all r, r' that agree up to time t:

$$(R, r, t) \models \varphi \quad \text{iff} \quad (R, r', t) \models \varphi$$

Past-dependent Formulas

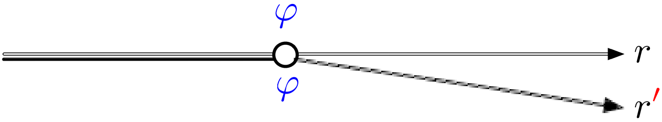


Definition

φ is past-dependent in R if for all r, r' that agree up to time t:

$$(R, r, t) \models \varphi \quad \text{iff} \quad (R, r', t) \models \varphi$$

Past-dependent Formulas



Definition

φ is past-dependent in R if for all r, r' that agree up to time t:

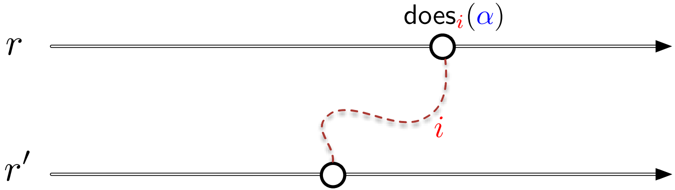
$$(R, r, t) \models \varphi \quad \text{iff} \quad (R, r', t) \models \varphi$$

State-enabled Actions

Definition

Action α is state-enabled in R when for all points (r, t) and (r', t') of R, if $r_i(t) = r'_i(t')$ & $(R, r, t) \models \mathsf{does}_i(\alpha)$ then there is a run r'' that agrees with r' up to time t' such that $(R, r'', t') \models \mathsf{does}_i(\alpha)$.

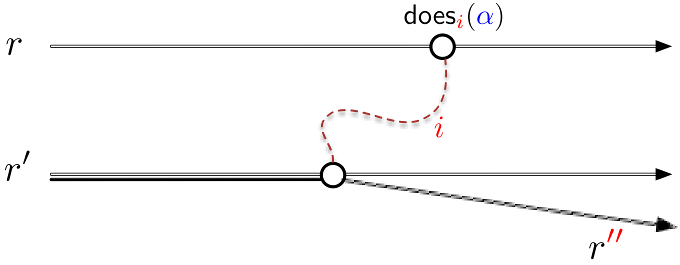
State-enabled Actions



Definition

Action α is state-enabled in R when for all points (r, t) and (r', t') of R, if $r_i(t) = r'_i(t')$ & $(R, r, t) \models \mathtt{does}_i(\alpha)$ then there is a run r'' that agrees with r' up to time t' such that $(R, r'', t') \models \mathtt{does}_i(\alpha)$.

State-enabled Actions



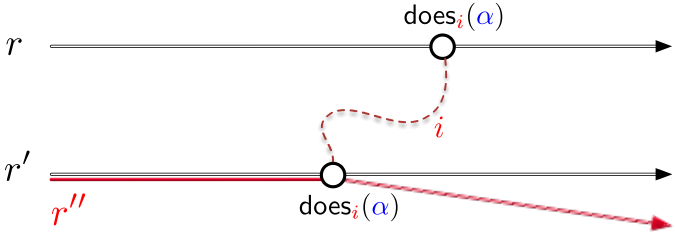
Definition

Action $\boldsymbol{\alpha}$ is state-enabled in R when for all points (r,t) and (r',t') of R, if

$r_i(t) = r'_i(t')$ & $(R,r,t) \models \mathtt{does}_i(\alpha)$ then there is a run r'' that agrees

with r' up to time t' such that $(R,r'',t') \models \mathtt{does}_i(\alpha)$.

State-enabled Actions



Definition

Action α is state-enabled in R when for all points (r,t) and (r',t') of R, if $r_i(t) = r'_i(t')$ & $(R,r,t) \models \texttt{does}_i(\alpha)$ then there is a run r'' that agrees with r' up to time t' such that $(R,r'',t') \models \texttt{does}_i(\alpha)$.

The **K*o*P** Theorem for Nondeterministic Actions

Theorem (Nondeterministic K*o*P)

Let α be state-enabled and φ be past-dependent in R.

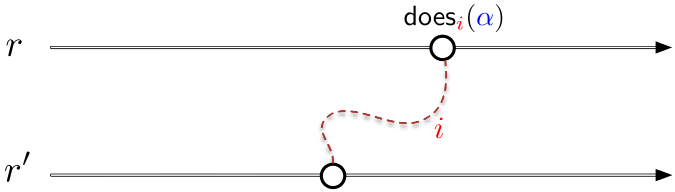
If φ *is a necessary condition for* $\texttt{does}_i(\alpha)$ *in R,*

then $K_i\varphi$ *is a necessary condition for* $\texttt{does}_i(\alpha)$ *in R.*

Proof of **K*o*P**

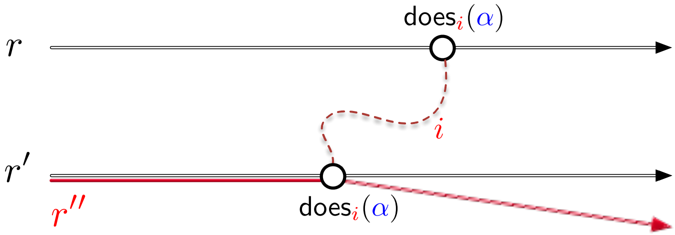
$$(R, r, t) \models \texttt{does}_i(\alpha)$$

Proof of **K*o*P**



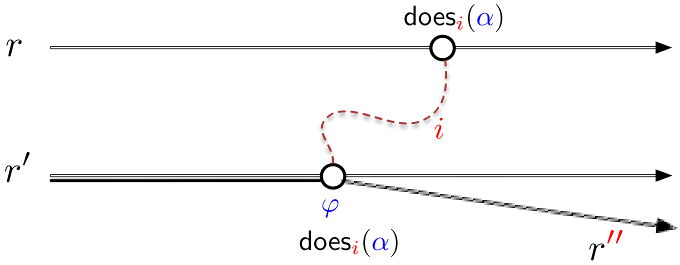
$$(r, t) \;\approx_i\; (r', t')$$

Proof of **K*o*P**



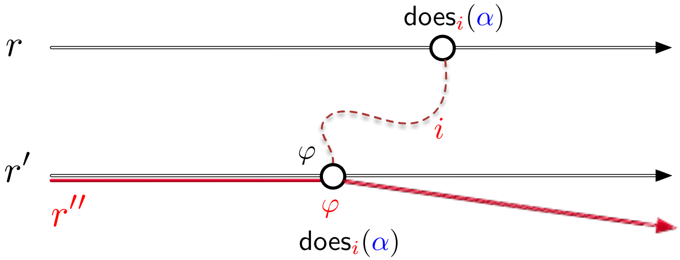
α is state-enabled

Proof of **K*o*P**



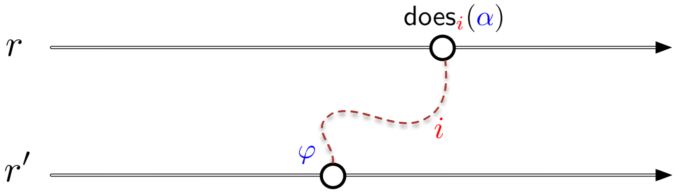
φ is a necessary condition

Proof of **K*o*P**



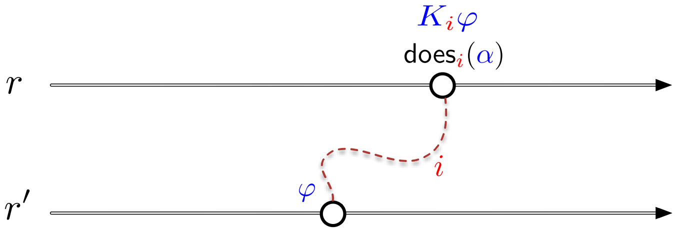
φ is past-dependent

Proof of KoP



φ holds at all indistinguishable points.

Proof of **K*o*P**



φ holds at all indistinguishable points.

Proof of **KoP**

$\mathtt{does}_i(\alpha) \;\Rightarrow\; K_i\varphi$ QED