The Coinductive Formulation of Common Knowledge

Colm Baston and Venanzio Capretta

Functional Programming Lab
School of Computer Science
University of Nottingham

ITP, Oxford, 09/07/2018

Epistemic Modal Logic

A propositional logic extended with a “knows” operator, written K, that represents the knowledge of an agent.

Epistemic Modal Logic

A propositional logic extended with a “knows” operator, written K, that represents the knowledge of an agent. The modal logic S5 provides rules for reasoning with knowledge operators:

Epistemic Modal Logic

A propositional logic extended with a “knows” operator, written K, that represents the knowledge of an agent. The modal logic S5 provides rules for reasoning with knowledge operators:

- Knowledge generalisation states that the agent can derive all tautologies.

Epistemic Modal Logic

A propositional logic extended with a “knows” operator, written K, that represents the knowledge of an agent. The modal logic S5 provides rules for reasoning with knowledge operators:

- Knowledge generalisation states that the agent can derive all tautologies.
- Axiom K states that the agent can follow implications, applying modus ponens to what they know.

Epistemic Modal Logic

A propositional logic extended with a “knows” operator, written K, that represents the knowledge of an agent. The modal logic S5 provides rules for reasoning with knowledge operators:

- Knowledge generalisation states that the agent can derive all tautologies.
- Axiom K states that the agent can follow implications, applying modus ponens to what they know.
- Axiom T states that the agent’s knowledge must actually be true, distinguishing knowledge from belief or opinion.

Epistemic Modal Logic

A propositional logic extended with a “knows” operator, written K, that represents the knowledge of an agent. The modal logic S5 provides rules for reasoning with knowledge operators:

- Knowledge generalisation states that the agent can derive all tautologies.
- Axiom K states that the agent can follow implications, applying modus ponens to what they know.
- Axiom T states that the agent’s knowledge must actually be true, distinguishing knowledge from belief or opinion.
- Axioms 4 and 5 state that the agent can perform introspection, knowing what they do and do not know.

Relational Semantics

The standard semantics for S5 interprets knowledge operators as equivalence relations over a set of possible worlds, which we call states:

Relational Semantics

The standard semantics for S5 interprets knowledge operators as equivalence relations over a set of possible worlds, which we call states:

- A state encodes all relevant information about the world we are modelling.

Relational Semantics

The standard semantics for S5 interprets knowledge operators as equivalence relations over a set of possible worlds, which we call states:

- A state encodes all relevant information about the world we are modelling.
- Epistemic propositions, which we call events, may be true in some states but false in others.

Relational Semantics

The standard semantics for S5 interprets knowledge operators as equivalence relations over a set of possible worlds, which we call states:

- A state encodes all relevant information about the world we are modelling.
- Epistemic propositions, which we call events, may be true in some states but false in others.
- Two states are related by an agent's knowledge relation iff they cannot distinguish one state from the other based on their knowledge.

Relational Semantics

The standard semantics for S5 interprets knowledge operators as equivalence relations over a set of possible worlds, which we call states:

- A state encodes all relevant information about the world we are modelling.
- Epistemic propositions, which we call events, may be true in some states but false in others.
- Two states are related by an agent's knowledge relation iff they cannot distinguish one state from the other based on their knowledge.
- The equivalence properties correspond to the S5 axioms:
  - Axiom T $\leftrightarrow$ Reflexivity
  - Axiom 4 $\leftrightarrow$ Transitivity
  - Axiom 5 $\leftrightarrow$ Transitivity and Symmetry

Shallow Embedding

We embed epistemic logic in type theory along these lines by postulating a set of states, and defining events as predicates over them:

$$\mathsf{State} : \mathsf{Set} \qquad\qquad \mathsf{Event} = \mathsf{State} \rightarrow \mathsf{Set}$$

Shallow Embedding

We embed epistemic logic in type theory along these lines by postulating a set of states, and defining events as predicates over them:

$$\text{State} : \text{Set} \qquad\qquad \text{Event} = \text{State} \rightarrow \text{Set}$$

$$\_ \sqcap \_ : \text{Event} \rightarrow \text{Event} \rightarrow \text{Event}$$
$$e_1 \sqcap e_2 = \lambda w. e_1\, w \wedge e_2\, w$$

$$\_ \sqcup \_ : \text{Event} \rightarrow \text{Event} \rightarrow \text{Event}$$
$$e_1 \sqcup e_2 = \lambda w. e_1\, w \vee e_2\, w$$

$$\_ \sqsubset \_ : \text{Event} \rightarrow \text{Event} \rightarrow \text{Event}$$
$$e_1 \sqsubset e_2 = \lambda w. e_1\, w \rightarrow e_2\, w$$

$$\sim \_ : \text{Event} \rightarrow \text{Event}$$
$$\sim e = \lambda w. \neg (e\, w)$$

Shallow Embedding

We embed epistemic logic in type theory along these lines by postulating a set of states, and defining events as predicates over them:

$$\mathsf{State} : \mathsf{Set} \qquad \mathsf{Event} = \mathsf{State} \to \mathsf{Set}$$

$$\_\sqcap\_ : \mathsf{Event} \to \mathsf{Event} \to \mathsf{Event}$$
$$e_1 \sqcap e_2 = \lambda w . e_1\, w \wedge e_2\, w$$

$$\_\sqcup\_ : \mathsf{Event} \to \mathsf{Event} \to \mathsf{Event}$$
$$e_1 \sqcup e_2 = \lambda w . e_1\, w \vee e_2\, w$$

$$\_\sqsubset\_ : \mathsf{Event} \to \mathsf{Event} \to \mathsf{Event}$$
$$e_1 \sqsubset e_2 = \lambda w . e_1\, w \to e_2\, w$$

$$\sim\_ : \mathsf{Event} \to \mathsf{Event}$$
$$\sim e = \lambda w . \neg(e\, w)$$

$$\_\subset\_ : \mathsf{Event} \to \mathsf{Event} \to \mathsf{Set}$$
$$e_1 \subset e_2 = \forall w . e_1\, w \to e_2\, w$$

$$\_\equiv\_ : \mathsf{Event} \to \mathsf{Event} \to \mathsf{Set}$$
$$e_1 \equiv e_2 = (e_1 \subset e_2) \wedge (e_2 \subset e_1)$$

$$\forall\!\!\!\forall : \mathsf{Event} \to \mathsf{Set}$$
$$\forall\!\!\!\forall\ e = \forall w . e\, w$$

Knowledge Operators

With these connectives, we can directly interpret the rules of S5 to define the concept of a knowledge operator:

Knowledge Operators

With these connectives, we can directly interpret the rules of S5 to define the concept of a knowledge operator:

$$
\begin{array}{l}
\text{Record KOp } (\mathsf{K} : \text{Event} \to \text{Event}) : \text{Set} \\
\quad \textit{generalisation} : \forall\!\!\!\forall\ e \to \forall\!\!\!\forall\ \mathsf{K}\, e \\
\quad \textit{axiomK} : \mathsf{K}\,(e_1 \sqsubset e_2) \subset (\mathsf{K}\, e_1 \sqsubset \mathsf{K}\, e_2) \\
\quad \textit{axiomT} : \mathsf{K}\, e \subset e \\
\quad \textit{axiom4} : \mathsf{K}\, e \subset \mathsf{K}\,(\mathsf{K}\, e) \\
\quad \textit{axiom5} : \sim\! \mathsf{K}\, e \subset \mathsf{K}\,(\sim\! \mathsf{K}\, e)
\end{array}
$$

Knowledge Operators

With these connectives, we can directly interpret the rules of S5 to define the concept of a knowledge operator:

$$
\begin{aligned}
&\mathsf{Record\ KOp\ (K : Event \to Event) : Set}\\
&\quad \mathit{generalisation} : \forall\!\!\!\forall\ e \to \forall\!\!\!\forall\ \mathsf{K}\ e\\
&\quad \mathit{axiomK} : \mathsf{K}\,(e_1 \sqsubset e_2) \subset (\mathsf{K}\ e_1 \sqsubset \mathsf{K}\ e_2)\\
&\quad \mathit{axiomT} : \mathsf{K}\ e \subset e\\
&\quad \mathit{axiom4} : \mathsf{K}\ e \subset \mathsf{K}\,(\mathsf{K}\ e)\\
&\quad \mathit{axiom5} : \sim \mathsf{K}\ e \subset \mathsf{K}\,(\sim \mathsf{K}\ e)
\end{aligned}
$$

But to prove the correspondence with the relational semantics, we had to add an infinitary deduction rule that allows agents to reason from a potentially infinite set of premises.

Event Families

An event family indexed by some type X is a function:

$$E : X \to \text{Event}$$

Event Families

An event family indexed by some type X is a function:

$$E : X \to \text{Event}$$

We can generate the event $\prod E$ that is true in those states where all events in the family E are true:

$$\prod : (X \to \text{Event}) \to \mathit{Event}$$
$$\prod E = \lambda w. \forall (x : X). E\, x\, w$$

Event Families

An event family indexed by some type X is a function:

$$E : X \to \text{Event}$$

We can generate the event $\prod E$ that is true in those states where all events in the family E are true:

$$\prod : (X \to \text{Event}) \to \mathit{Event}$$
$$\prod E = \lambda w. \forall (x : X). E\, x\, w$$

We can map a knowledge operator K onto the whole family by applying it to every member. We just write this as $\mathsf{K}\, E$:

$$\mathsf{K}\, E := \lambda x. \mathsf{K}\, (E\, x)$$

Preservation of Semantic Entailment

We say that K preserves semantic entailment iff for every event family E and event e:

$$\prod E \subset e \rightarrow \prod(K\ E) \subset K\ e$$

Preservation of Semantic Entailment

We say that K preserves semantic entailment iff for every event family E and event e:

$$\prod E \subset e \rightarrow \prod (K\ E) \subset K\ e$$

In addition to the previous properties, we require that knowledge operators preserve semantic entailment.

Preservation of Semantic Entailment

We say that K preserves semantic entailment iff for every event family E and event e:

$$\prod E \subset e \to \prod (K\ E) \subset K\ e$$

In addition to the previous properties, we require that knowledge operators preserve semantic entailment. In fact, knowledge generalisation and Axiom K are special cases of this:

Preservation of Semantic Entailment

We say that K preserves semantic entailment iff for every event family E and event e:

$$\bigsqcap E \subset e \rightarrow \bigsqcap(K\ E) \subset K\ e$$

In addition to the previous properties, we require that knowledge operators preserve semantic entailment. In fact, knowledge generalisation and Axiom K are special cases of this:

- For knowledge generalisation, observe that $\forall\!\!\!\forall\ e$ is equivalent to semantic entailment from the empty family: $\bigsqcap \varnothing \subset e$.

Preservation of Semantic Entailment

We say that K preserves semantic entailment iff for every event family E and event e:

$$\prod E \subset e \rightarrow \prod (K\ E) \subset K\ e$$

In addition to the previous properties, we require that knowledge operators preserve semantic entailment. In fact, knowledge generalisation and Axiom K are special cases of this:

- For knowledge generalisation, observe that $\forall\!\!\!\forall\ e$ is equivalent to semantic entailment from the empty family: $\prod \varnothing \subset e$.
- For Axiom K, choose a "modus ponens" family indexed by the Booleans: $\prod \{e_1 \sqsubset e_2, e_1\} \subset e_2$.

Transformations

To prove the correspondence between the knowledge operator and relational semantics, we define transformations between the two:

Transformations

To prove the correspondence between the knowledge operator and relational semantics, we define transformations between the two:

$$\mathsf{K}_{[\,]} : (\mathsf{State} \to \mathsf{State} \to \mathsf{Set}) \to (\mathsf{Event} \to \mathsf{Event})$$
$$\mathsf{K}_{[R]} = \lambda e. \lambda w. \forall v. w\, R\, v \to e\, v$$

Transformations

To prove the correspondence between the knowledge operator and relational semantics, we define transformations between the two:

$$\mathsf{K}_{[\,]} : (\mathsf{State} \to \mathsf{State} \to \mathsf{Set}) \to (\mathsf{Event} \to \mathsf{Event})$$
$$\mathsf{K}_{[R]} = \lambda e.\lambda w.\forall v.w\, R\, v \to e\, v$$

$$R_{[\,]} : (\mathsf{Event} \to \mathsf{Event}) \to (\mathsf{State} \to \mathsf{State} \to \mathsf{Set})$$
$$R_{[\mathsf{K}]} = \lambda w.\lambda v.\forall e.\mathsf{K}\, e\, w \to \mathsf{K}\, e\, v$$

Transformations

To prove the correspondence between the knowledge operator and relational semantics, we define transformations between the two:

$$\mathsf{K}_{[\ ]} : (\mathsf{State} \to \mathsf{State} \to \mathsf{Set}) \to (\mathsf{Event} \to \mathsf{Event})$$
$$\mathsf{K}_{[R]} = \lambda e.\lambda w.\forall v.w\, R\, v \to e\, v$$

$$R_{[\ ]} : (\mathsf{Event} \to \mathsf{Event}) \to (\mathsf{State} \to \mathsf{State} \to \mathsf{Set})$$
$$R_{[\mathsf{K}]} = \lambda w.\lambda v.\forall e.\mathsf{K}\, e\, w \to \mathsf{K}\, e\, v$$

- The fact that applying $\mathsf{K}_{[\ ]}$ to an equivalence relation results in a operator satisfying S5 is well-known in the literature. We additionally had to prove preservation of semantic entailment.

Transformations

To prove the correspondence between the knowledge operator and relational semantics, we define transformations between the two:

$$\mathsf{K}_{[\ ]} : (\mathsf{State} \to \mathsf{State} \to \mathsf{Set}) \to (\mathsf{Event} \to \mathsf{Event})$$
$$\mathsf{K}_{[R]} = \lambda e.\lambda w.\forall v.w\, R\, v \to e\, v$$

$$R_{[\ ]} : (\mathsf{Event} \to \mathsf{Event}) \to (\mathsf{State} \to \mathsf{State} \to \mathsf{Set})$$
$$R_{[\mathsf{K}]} = \lambda w.\lambda v.\forall e.\mathsf{K}\, e\, w \to \mathsf{K}\, e\, v$$

- The fact that applying $\mathsf{K}_{[\ ]}$ to an equivalence relation results in a operator satisfying S5 is well-known in the literature. We additionally had to prove preservation of semantic entailment.
- The inverse proof requires use of Axiom 5 and classical logic to show symmetry.

Isomorphism

To prove that the transformations are in fact inverse, we must show:

$$\mathsf{K}_{[R_{[\mathsf{K}]}]}\, e\, w \leftrightarrow \mathsf{K}\, e\, w \qquad \textit{and} \qquad R_{[\mathsf{K}_{[R]}]}\, w\, v \leftrightarrow R\, w\, v$$

Isomorphism

To prove that the transformations are in fact inverse, we must show:

$$\mathsf{K}_{[R_{[\mathsf{K}]}]}\, e\, w \leftrightarrow \mathsf{K}\, e\, w \qquad \textit{and} \qquad R_{[\mathsf{K}_{[R]}]}\, w\, v \leftrightarrow R\, w\, v$$

Three directions are straightforward, but the fourth uses preservation of semantic entailment.

Isomorphism

To prove that the transformations are in fact inverse, we must show:

$$\mathsf{K}_{[R_{[\mathsf{K}]}]}\, e\, w \leftrightarrow \mathsf{K}\, e\, w \qquad \textit{and} \qquad R_{[\mathsf{K}_{[R]}]}\, w\, v \leftrightarrow R\, w\, v$$

Three directions are straightforward, but the fourth uses preservation of semantic entailment. For the proof, we characterise $\mathsf{K}_{[R_{[\mathsf{K}]}]}\, e\, w$ using an event family:

$$\begin{aligned} &\mathsf{KFam}^{w} : (\Sigma e.\mathsf{K}\, e\, w) \to \mathsf{Event} \\ &\mathsf{KFam}^{w}\, \langle e, \_ \rangle = \mathsf{K}\, e \end{aligned}$$

Isomorphism

To prove that the transformations are in fact inverse, we must show:

$$\mathsf{K}_{[R_{[\mathsf{K}]}]}\, e\, w \leftrightarrow \mathsf{K}\, e\, w \qquad \textit{and} \qquad R_{[\mathsf{K}_{[R]}]}\, w\, v \leftrightarrow R\, w\, v$$

Three directions are straightforward, but the fourth uses preservation of semantic entailment. For the proof, we characterise $\mathsf{K}_{[R_{[\mathsf{K}]}]}\, e\, w$ using an event family:

$$\begin{aligned} &\mathsf{KFam}^{w} : (\Sigma e. \mathsf{K}\, e\, w) \to \mathsf{Event} \\ &\mathsf{KFam}^{w}\, \langle e, \_ \rangle = \mathsf{K}\, e \end{aligned}$$

By unfolding the definitions of the transformations, we find that:

$$\mathsf{K}_{[R_{[\mathsf{K}]}]}\, e\, w \to \prod(\mathsf{KFam}^{w}) \subset e$$

Isomorphism

We can replace the premise of the fourth direction using the previous fact, leaving us to prove $\mathsf{K}\, e\, w$ from the assumption:

$$\prod(\mathsf{KFam}^{w}) \subset e$$

Isomorphism

We can replace the premise of the fourth direction using the previous fact, leaving us to prove $\mathsf{K}\, e\, w$ from the assumption:

$$\prod(\mathsf{KFam}^{w}) \subset e$$

Applying preservation of semantic entailment and instantiating at state w:

$$\prod(\mathsf{K}\, \mathsf{KFam}^{w})\, w \to \mathsf{K}\, e\, w$$

Isomorphism

We can replace the premise of the fourth direction using the previous fact, leaving us to prove $\mathsf{K}\, e\, w$ from the assumption:

$$\prod(\mathsf{KFam}^w) \subset e$$

Applying preservation of semantic entailment and instantiating at state w:

$$\prod(\mathsf{K}\,\mathsf{KFam}^w)\, w \to \mathsf{K}\, e\, w$$

So we now need to show that all elements of family $\mathsf{K}\,\mathsf{KFam}^w$ are true in state w:

Isomorphism

We can replace the premise of the fourth direction using the previous fact, leaving us to prove $\mathsf{K}\,e\,w$ from the assumption:

$$\prod(\mathsf{KFam}^w) \subset e$$

Applying preservation of semantic entailment and instantiating at state w:

$$\prod(\mathsf{K}\,\mathsf{KFam}^w)\,w \to \mathsf{K}\,e\,w$$

So we now need to show that all elements of family $\mathsf{K}\,\mathsf{KFam}^w$ are true in state w:

- KFam^w indices are of the form $\langle e', h\rangle$, where h is a proof that $K\ e'\ w$, and the element at that index is $\mathsf{K}\,e'$.

Isomorphism

We can replace the premise of the fourth direction using the previous fact, leaving us to prove $\mathsf{K}\, e\, w$ from the assumption:

$$\prod(\mathsf{KFam}^{w}) \subset e$$

Applying preservation of semantic entailment and instantiating at state w:

$$\prod(\mathsf{K}\,\mathsf{KFam}^{w})\, w \to \mathsf{K}\, e\, w$$

So we now need to show that all elements of family $\mathsf{K}\,\mathsf{KFam}^{w}$ are true in state w:

- KFam^{w} indices are of the form $\langle e', h\rangle$, where h is a proof that $K\ e'\ w$, and the element at that index is $\mathsf{K}\, e'$.
- We have mapped K onto the family, so we must actually prove $\mathsf{K}\,(\mathsf{K}\, e')$ at state w.

Isomorphism

We can replace the premise of the fourth direction using the previous fact, leaving us to prove $\mathsf{K}\,e\,w$ from the assumption:

$$\prod(\mathsf{KFam}^w) \subset e$$

Applying preservation of semantic entailment and instantiating at state w:

$$\prod(\mathsf{K}\,\mathsf{KFam}^w)\,w \to \mathsf{K}\,e\,w$$

So we now need to show that all elements of family $\mathsf{K}\,\mathsf{KFam}^w$ are true in state w:

- KFam^w indices are of the form $\langle e', h \rangle$, where h is a proof that $K\ e'\ w$, and the element at that index is $\mathsf{K}\,e'$.
- We have mapped K onto the family, so we must actually prove $\mathsf{K}\,(\mathsf{K}\,e')$ at state w.
- We can conclude this by applying Axiom 4 to h.

Group Knowledge

Epistemic logic can be extended to a group of agents, allowing for several forms of group knowledge:

Group Knowledge

Epistemic logic can be extended to a group of agents, allowing for several forms of group knowledge:

- An event is universal knowledge iff every agent knows it.

Group Knowledge

Epistemic logic can be extended to a group of agents, allowing for several forms of group knowledge:

- An event is universal knowledge iff every agent knows it.
- An event is distributed knowledge iff it is derivable from the total pool of all agents' knowledge.

Group Knowledge

Epistemic logic can be extended to a group of agents, allowing for several forms of group knowledge:

- An event is universal knowledge iff every agent knows it.
- An event is distributed knowledge iff it is derivable from the total pool of all agents' knowledge.
- An event is common knowledge iff every agent knows it, every agent knows that every agent knows it, and so on ad infinitum.

Group Knowledge

Epistemic logic can be extended to a group of agents, allowing for several forms of group knowledge:

- An event is universal knowledge iff every agent knows it.
- An event is distributed knowledge iff it is derivable from the total pool of all agents' knowledge.
- An event is common knowledge iff every agent knows it, every agent knows that every agent knows it, and so on ad infinitum.

From this point on, we postulate a non-empty set of agents and equip each $a \in \mathsf{Agent}$ with their own knowledge relation $\simeq_a$. This also provides each with a knowledge operator $\mathsf{K}_a = \mathsf{K}_{[\simeq_a]}$.

Relational Common Knowledge

Common knowledge is defined in the relational semantics by its own equivalence relation, which we write $\propto$:

Relational Common Knowledge

Common knowledge is defined in the relational semantics by its own equivalence relation, which we write $\propto$:

$$\text{Inductive } \_ \propto \_ : \text{State} \rightarrow \text{State} \rightarrow \text{Set}$$
$$\propto\!-\text{union} : \forall a.\forall w.\forall v.w \simeq_a v \rightarrow w \propto v$$
$$\propto\!-\text{trans} : \forall w.\forall v.\forall u.w \propto v \rightarrow v \propto u \rightarrow w \propto u$$

Relational Common Knowledge

Common knowledge is defined in the relational semantics by its own equivalence relation, which we write $\propto$:

$$\begin{aligned}
&\text{Inductive } _ \propto _ : \text{State} \to \text{State} \to \text{Set} \\
&\qquad \propto\!-\text{union} : \forall a.\forall w.\forall v.w \simeq_a v \to w \propto v \\
&\qquad \propto\!-\text{trans} : \forall w.\forall v.\forall u.w \propto v \to v \propto u \to w \propto u
\end{aligned}$$

- This is the transitive closure of the union of all agents' knowledge relations.

Relational Common Knowledge

Common knowledge is defined in the relational semantics by its own equivalence relation, which we write $\propto$:

$$\text{Inductive } \_ \propto \_ : \text{State} \to \text{State} \to \text{Set}$$
$$\propto\!-\text{union} : \forall a. \forall w. \forall v. w \simeq_a v \to w \propto v$$
$$\propto\!-\text{trans} : \forall w. \forall v. \forall u. w \propto v \to v \propto u \to w \propto u$$

- This is the transitive closure of the union of all agents' knowledge relations.
- It can be proved to be an equivalence relation using the fact that each $\simeq_a$ is itself an equivalence relation.

Relational Common Knowledge

Common knowledge is defined in the relational semantics by its own equivalence relation, which we write $\propto$:

$$\begin{aligned}&\text{Inductive} \_ \propto \_ : \text{State} \to \text{State} \to \text{Set}\\ &\quad \propto\!-\text{union} : \forall a.\forall w.\forall v.w \simeq_a v \to w \propto v\\ &\quad \propto\!-\text{trans} : \forall w.\forall v.\forall u.w \propto v \to v \propto u \to w \propto u\end{aligned}$$

- This is the transitive closure of the union of all agents' knowledge relations.
- It can be proved to be an equivalence relation using the fact that each $\simeq_a$ is itself an equivalence relation.

Since it is an equivalence relation, we can generate a common knowledge operator from it:

$$\begin{aligned}&\text{rCK} : \text{Event} \to \text{Event}\\ &\text{rCK} = \text{K}_{[\propto]}\end{aligned}$$

Coinductive Common Knowledge

Defining a universal knowledge operator "everyone knows":

$$\mathsf{EK} : \mathsf{Event} \rightarrow \mathsf{Event}$$
$$\mathsf{EK}\, e = \lambda w. \forall a. \mathsf{K}_a\, e\, w$$

Coinductive Common Knowledge

Defining a universal knowledge operator "everyone knows":

$$\mathsf{EK} : \mathsf{Event} \rightarrow \mathsf{Event}$$
$$\mathsf{EK}\, e = \lambda w. \forall a. \mathsf{K}_a\, e\, w$$

Following the informal description, we can see common knowledge of an event e as the infinite conjunction:

$$EK\; e \sqcap EK\,(EK\; e) \sqcap EK\,(EK\,(EK\; e)) \sqcap \ldots$$

Coinductive Common Knowledge

Defining a universal knowledge operator "everyone knows":

$$\begin{aligned} &\mathsf{EK} : \mathsf{Event} \to \mathsf{Event} \\ &\mathsf{EK}\, e = \lambda w. \forall a. \mathsf{K}_a\, e\, w \end{aligned}$$

Following the informal description, we can see common knowledge of an event e as the infinite conjunction:

$$EK\ e \sqcap EK\,(EK\ e) \sqcap EK\,(EK\,(EK\ e)) \sqcap \ldots$$

This leads naturally to a coinductive definition:

$$\begin{aligned} &\mathsf{CoInductive}\ \mathsf{cCK} : \mathsf{Event} \to \mathsf{Event} \\ &\qquad \mathsf{cCK}{-}\mathsf{intro} : \mathsf{EK} e \sqcap \mathsf{cCK}\,(\mathsf{EK}\, e) \subset \mathsf{cCK}\, e \end{aligned}$$

Coinductive Common Knowledge

The advantages of the coinductive approach are:

Coinductive Common Knowledge

The advantages of the coinductive approach are:

- It more-closely matches the intuitive description as an infinite conjunction of events.

Coinductive Common Knowledge

The advantages of the coinductive approach are:

- It more-closely matches the intuitive description as an infinite conjunction of events.
- It can be formulated at a higher level, hiding the underlying use of states with the connectives that we have defined.

Coinductive Common Knowledge

The advantages of the coinductive approach are:

- It more-closely matches the intuitive description as an infinite conjunction of events.
- It can be formulated at a higher level, hiding the underlying use of states with the connectives that we have defined.
- It provides us a new tool for reasoning about common knowledge: guarded corecursion.

Coinductive Common Knowledge

The advantages of the coinductive approach are:

- It more-closely matches the intuitive description as an infinite conjunction of events.
- It can be formulated at a higher level, hiding the underlying use of states with the connectives that we have defined.
- It provides us a new tool for reasoning about common knowledge: guarded corecursion.

However, we still need to establish that cCK is in fact equivalent to the relational commmon knowledge operator. That is, for all events e:

$$\mathsf{rCK}\ e \equiv \mathsf{cCK}\ e$$

Common Knowledge Equivalence

We will only show the left-to-right direction here.

Common Knowledge Equivalence

We will only show the left-to-right direction here. We first prove two lemmas about rCK, it implies statements that correspond to the fields of constructor cCK−intro:

- $\mathsf{rCK}\, e \subset \mathsf{EK}\, e$
- $\mathsf{rCK}\, e \subset \mathsf{rCK}\, (\mathsf{EK}\, e)$

Common Knowledge Equivalence

We will only show the left-to-right direction here. We first prove two lemmas about rCK, it implies statements that correspond to the fields of constructor cCK−intro:

- $\mathsf{rCK}\, e \subset \mathsf{EK}\, e$
- $\mathsf{rCK}\, e \subset \mathsf{rCK}\,(\mathsf{EK}\, e)$

The proofs of these are by fully unfolding the definitions of rCK, EK, and K_a until we are working directly with the underlying relations:

- $\forall w.(\forall v.w \propto v \to e\, v) \to \forall a.\forall u.w \simeq_a u \to e\, u$
- $\forall w.(\forall v.w \propto v \to e\, v) \to \forall u.w \propto u \to \forall a.\forall t.u \simeq_a t \to e\, t$

Common Knowledge Equivalence

We will only show the left-to-right direction here. We first prove two lemmas about rCK, it implies statements that correspond to the fields of constructor cCK−intro:

- $\mathsf{rCK}\, e \subset \mathsf{EK}\, e$
- $\mathsf{rCK}\, e \subset \mathsf{rCK}\,(\mathsf{EK}\, e)$

The proofs of these are by fully unfolding the definitions of rCK, EK, and K_a until we are working directly with the underlying relations:

- $\forall w.(\forall v.w \propto v \to e\, v) \to \forall a.\forall u.w \simeq_a u \to e\, u$
- $\forall w.(\forall v.w \propto v \to e\, v) \to \forall u.w \propto u \to \forall a.\forall t.u \simeq_a t \to e\, t$

Then the assumptions can be combined with the constructors of $\propto$ to reach the desired conclusions.

Common Knowledge Equivalence

The rest of the proof proceeds by coinduction:

Common Knowledge Equivalence

The rest of the proof proceeds by coinduction:

- The statement we are to proving is: $\forall e.\, \mathsf{rCK}\, e \subset \mathsf{cCK}\, e$.

Common Knowledge Equivalence

The rest of the proof proceeds by coinduction:

- The statement we are to proving is: $\forall e.\, \mathsf{rCK}\, e \subset \mathsf{cCK}\, e$. We are allowed to assume it as a coinductive hypothesis provided that we use it only when guarded by constructor $\mathsf{cCK}-\mathsf{intro}$.

Common Knowledge Equivalence

The rest of the proof proceeds by coinduction:

- The statement we are to proving is: $\forall e.\, \mathsf{rCK}\, e \subset \mathsf{cCK}\, e$. We are allowed to assume it as a coinductive hypothesis provided that we use it only when guarded by constructor $\mathsf{cCK}-\mathsf{intro}$.
- We assume $\mathsf{rCK}\, e$ and apply $\mathsf{cCK}-\mathsf{intro}$, leaving us with two proof obligations: $\mathsf{EK}\, e$ and $\mathsf{cCK}\, (\mathsf{EK}\, e)$.

Common Knowledge Equivalence

The rest of the proof proceeds by coinduction:

- The statement we are to proving is: $\forall e.\, \mathsf{rCK}\, e \subset \mathsf{cCK}\, e$. We are allowed to assume it as a coinductive hypothesis provided that we use it only when guarded by constructor $\mathsf{cCK}-\mathsf{intro}$.
- We assume $\mathsf{rCK}\, e$ and apply $\mathsf{cCK}-\mathsf{intro}$, leaving us with two proof obligations: $\mathsf{EK}\, e$ and $\mathsf{cCK}\,(\mathsf{EK}\, e)$.
- For $\mathsf{EK}\, e$, we simply use the first of the previous lemmas with our assumption $\mathsf{rCK}\, e$.

Common Knowledge Equivalence

The rest of the proof proceeds by coinduction:

- The statement we are to proving is: $\forall e.\, \mathsf{rCK}\, e \subset \mathsf{cCK}\, e$. We are allowed to assume it as a coinductive hypothesis provided that we use it only when guarded by constructor $\mathsf{cCK-intro}$.
- We assume $\mathsf{rCK}\, e$ and apply $\mathsf{cCK-intro}$, leaving us with two proof obligations: $\mathsf{EK}\, e$ and $\mathsf{cCK}\,(\mathsf{EK}\, e)$.
- For $\mathsf{EK}\, e$, we simply use the first of the previous lemmas with our assumption $\mathsf{rCK}\, e$.
- For $\mathsf{cCK}\,(\mathsf{EK}\, e)$, we use the second of the previous lemmas, deriving $\mathsf{rCK}\,(\mathsf{EK}\, e)$.

Common Knowledge Equivalence

The rest of the proof proceeds by coinduction:

- The statement we are to proving is: $\forall e.\, \mathsf{rCK}\, e \subset \mathsf{cCK}\, e$. We are allowed to assume it as a coinductive hypothesis provided that we use it only when guarded by constructor $\mathsf{cCK}-\mathsf{intro}$.
- We assume $\mathsf{rCK}\, e$ and apply $\mathsf{cCK}-\mathsf{intro}$, leaving us with two proof obligations: $\mathsf{EK}\, e$ and $\mathsf{cCK}\, (\mathsf{EK}\, e)$.
- For $\mathsf{EK}\, e$, we simply use the first of the previous lemmas with our assumption $\mathsf{rCK}\, e$.
- For $\mathsf{cCK}\, (\mathsf{EK}\, e)$, we use the second of the previous lemmas, deriving $\mathsf{rCK}\, (\mathsf{EK}\, e)$.
- We can then instantiate our coinductive hypothesis with the event $\mathsf{EK}\, e$, and apply it to the intermediate result above, concluding $\mathsf{cCK}\, (\mathsf{EK}\, e)$.

Conclusion

We have:

- Formalised an epistemic logic framework in type theory using a shallow embedding.

Conclusion

We have:

- Formalised an epistemic logic framework in type theory using a shallow embedding.
- Proved the equivalence of the knowledge operator and relational semantics in this framework using the new property of preservation of semantic entailment.

Conclusion

We have:

- Formalised an epistemic logic framework in type theory using a shallow embedding.
- Proved the equivalence of the knowledge operator and relational semantics in this framework using the new property of preservation of semantic entailment.
- Defined common knowledge using a coinductive data type.

Conclusion

We have:

- Formalised an epistemic logic framework in type theory using a shallow embedding.
- Proved the equivalence of the knowledge operator and relational semantics in this framework using the new property of preservation of semantic entailment.
- Defined common knowledge using a coinductive data type.
- Proved that this coinductive common knowledge operator coincides with the one generated by the relational semantics.

Conclusion

We have:

- Formalised an epistemic logic framework in type theory using a shallow embedding.
- Proved the equivalence of the knowledge operator and relational semantics in this framework using the new property of preservation of semantic entailment.
- Defined common knowledge using a coinductive data type.
- Proved that this coinductive common knowledge operator coincides with the one generated by the relational semantics.

Thank you!