Using Counterfactuals in Knowledge-Based Programming

Joe Halpern
Cornell University

Yoram Moses
Technion

Knowledge-Based Programs

Knowledge-based (kb) programs [Halpern-Fagin, 1985] provide a high-level way of designing and specifying protocols, by allowing tests on a user's knowledge:

$$\textbf{if } K(x = 0) \textbf{ then } y := y + 1 \textbf{ else } \mathsf{skip},$$

- If you know that $x = 0$ then set y to $y + 1$; else do nothing

Knowledge-based programs are useful [lots of examples in paper], but can sometimes behave in a counterintuitive way ...

The Bit-Transmission Problem

Suppose a sender S wants to communicate a bit to a receiver R over a possibly faulty communication line.

- S sends bit to R repeatedly until it receives an *ack*:

$$\textbf{if } \mathit{recack} \textbf{ then } \textsf{skip} \textbf{ else } \textsf{sendbit}$$

We can capture these intuitions using a kb program:

$$\textbf{if } K_S \mathit{recbit} \textbf{ then } \textsf{skip} \textbf{ else } \textsf{sendbit}$$

- If the sender knows that the receiver has received the bit, then it halts; otherwise it resends the bit.

Can further abstract away *how* knowledge is obtained:

$$\textbf{if } K_S K_R(\mathit{bit}) \textbf{ then } \textsf{skip} \textbf{ else } \textsf{sendbit}$$

- If the sender knows that the receiver knows the bit, then it halts; otherwise it resends the bit.
  - $K_R(\mathit{bit}) = K_R(\mathit{bit} = 0) \vee K_R(\mathit{bit} = 1)$.
  - the sender may know that the receiver knows the bit even without an acknowledgement

Further Optimizations

Suppose that messages are guaranteed to arrive within 5 rounds. Then

$$\textbf{if } K_S K_R(bit) \textbf{ then } \mathsf{skip} \textbf{ else } \mathsf{sendbit}$$

is "wasteful".

- The bit is sent 5 times; it suffices to send it once.

The obvious improvement:

$$\textbf{if } K_S \Diamond recbit \textbf{ then } \mathsf{skip} \textbf{ else } \mathsf{sendbit}$$

But there's a problem:

- Should S send the bit?
  - In systems where S sends the bit, $K_S \Diamond recbit$ always holds, so S shouldn't send the bit.
  - In systems where S doesn't send the bit, $K_S \Diamond recbit$ never holds, so S should send the bit.

Conclusion: S should send the bit iff S doesn't send it.

- This kb program is not implementable!

The same difficulties arise with

$$\textbf{if } K_S K_R(bit) \textbf{ then } \mathsf{skip} \textbf{ else } \mathsf{sendbit}$$

A Slightly Different Intuition

Instead of saying

- S should stop sending if S knows that R will eventually receive the bit,

we should say

- S should stop sending if it knows that *even if* S *does not send another message* R will eventually receive the bit.

Suppose $do(i, \mathsf{a})$ is true if process i performs a in the next round. Could try

$$\textbf{if } K_S(do(S, \mathsf{skip}) \Rightarrow \Diamond recbit) \textbf{ then } \mathsf{skip} \textbf{ else } \mathsf{sendbit}$$

- This has the same problems as the earlier program
- S should send the bit iff S doesn't send the bit

Counterfactuals to the Rescue

$\Rightarrow$ is a *material implication*

- $p \Rightarrow q$ is vacuously true if p is false

We need to use a *counterfactual implication* $p > q$:

- Suppose the match is wet. Is the statement "If the match were dry then it would light" true?
- How about "If the brakes weren't faulty, then I wouldn't have had the accident"
  - (even though I was drunk and it was pouring rain)

[Stalnaker, Lewis]: $p > q$ is true at a world w if q is true at the closest worlds to w where p is true.

- But what are the closest worlds?
  - A major problem for philosophers
  - Somewhat easier in the context of protocols

Another Complication

When does protocol P *implement* kb program Pg_{kb}?

- Idea: consider set $\mathcal{R}(P)$ of runs generated by P
- Evaluate knowledge tests in Pg_{kb} with respect to $\mathcal{R}(P)$
- See if Pg_{kb} then generates $\mathcal{R}(P)$

Problem: to evaluate counterfactuals, we need to consider runs *not* in $\mathcal{R}(P)$.

- These are runs counter to fact, where P is not followed
  - What would happen if the message weren't set (even though, according to P, it is)

Using a larger set of runs means we can no longer express global properties of P.

- Properties that hold in all runs of $\mathcal{R}(P)$.

Belief

Solution: consider *belief* instead of knowledge.

- Evaluate tests with respect to larger system that includes the counterfactual runs.
- Each run is *ranked*
  - Lower rank = more likely
  - Runs of P get rank 0; all other runs get higher rank
- A formula is *believed* if it holds in all the runs of lowest rank considered possible.
- This notion of belief coincides with knowledge when restricted to the runs of P

Using the counterfactual operator and this interpretation for belief, we get the program $\mathsf{BT}^{>}$:

$$\textbf{if } B_S(do(S, \mathsf{skip}) > \Diamond recbit) \textbf{ then } \mathsf{skip} \textbf{ else } \mathsf{sendbit}.$$

This program does what we want.

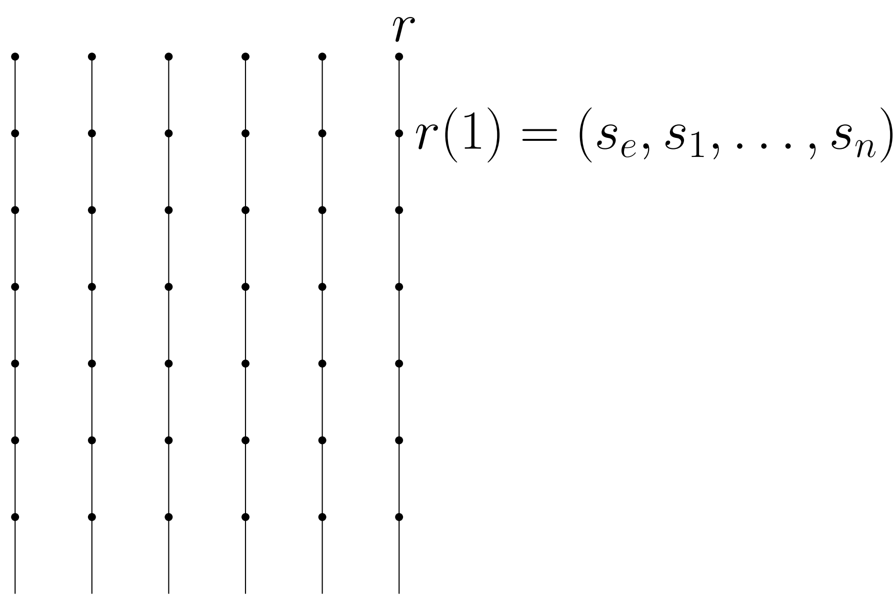
- The rest of the talk makes all this precise ...

Multi-Agent Systems

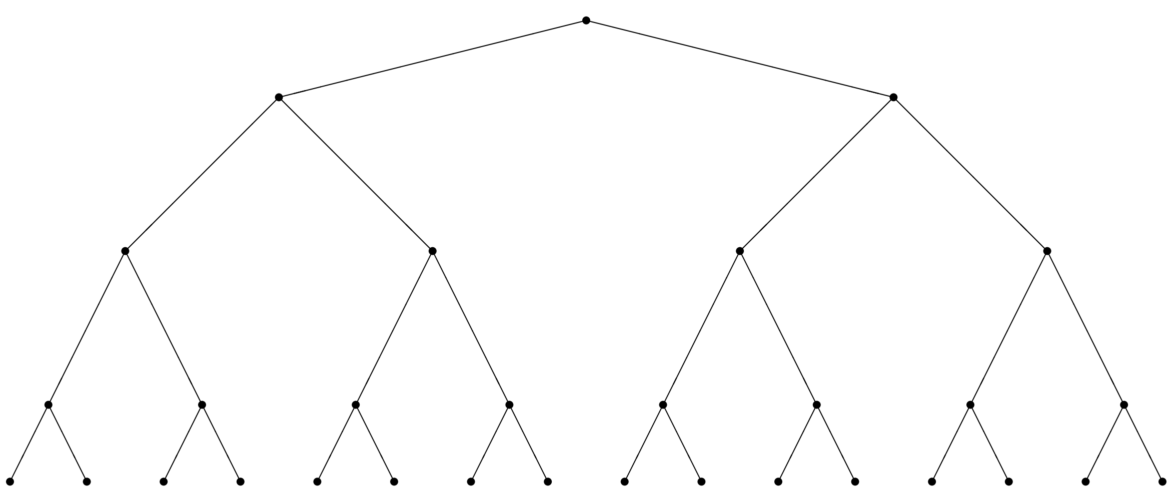
A system consists of a collection of processes/agents connected by a communication network

- Each agent has a *local* state
  - depends on initial state, messages received, etc.
  - captures all the information the agent can access.
- The *environment* state captures everything else that's relevant
- The *global state* is a tuple consisting of each process' (local) state + environment state
- A *run* of the system is a complete description of the system over time:
  - a function from times to global states
- A *system* is a set of runs
  - we identify a system with its possible behaviors

A system:



The runs in a system are often best thought of as the branches of a computation tree:



Knowledge in multi-agent systems

A system is a Kripke structure!

- The possible worlds are pairs (r, m)
  - Worlds now have structure (global states)
- $(r, m) \sim_i (r', m')$ if agent i has the same local state in $r(m)$ and $r'(m')$
  - $\sim_i$ defines an equivalence relation $\mathcal{K}_i$
- Also need an interpretation π

Let $\mathcal{I} = (\mathcal{R}, \pi)$

- $(\mathcal{I}, r, m) \models K_i\varphi$ if $(\mathcal{I}, r', m') \models \varphi$ for all (r', m') such that $r'_i(m') = r_i(m)$.

This is an idealized notion of knowledge, that does not take computation into account.

- It is still a useful tool for analyzing systems.

Protocols

Where do the runs in a system come from?

Typically they are generated by a *protocol*.

- a description of each agent's actions as a function of his local state
  - **if** receive message
    then send acknowledgement

Given a protocol P, can consider the sytem $\mathcal{R}(P, \gamma)$ consisting of all runs of P in *context* γ.

- Context specifies
  - whether messages are guaranteed to arrive
  - upper bounds on message delivery time
  - what type of faulty behavior is possible
  - ...

Knowledge-Based Programs

A kb program Pg_{kb} and an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ determine a protocol $\mathsf{Pg}_{kb}^{\mathcal{I}}$ in context γ:

- What does the protocol $\mathsf{Pg}_{kb}^{\mathcal{I}}$ do at point (r, m)?
  - If Pg_{kb} says "**if** $K\varphi$ **then** a **else** b", then evaluate whether $K_i\varphi$ is true at $(\mathcal{I}, r, m)$.

Does $\mathsf{Pg}_{kb}^{\mathcal{I}}$ generate $\mathcal{R}$?

- If so, protocol $\mathsf{Pg}_{kb}^{\mathcal{I}}$ *implements* Pg_{kb}.

A kb program may have 0, 1, or many protocols that implement it.

- Can think of a kb program as a *specification*
- Exist sufficient conditions for when a kb-program is implemented by a unique protocol.
- **if** $K_S\Diamond \mathit{recbit}$ **then** skip **else** $\mathsf{sendbit}$ is not implemented by any protocol

Interpreting Counterfactuals

A *counterfactual system* has the form $\mathcal{J} = (\mathcal{I}, \ll)$

- $\mathcal{I}$ is an interpreted system
- $\ll$ is an *order assignment*
  - $(r_1, m_1) \ll_{(r,m)} (r_2, m_2)$ means "(r_1, m_1) is closer to (r, m) than (r_2, m_2)"

$(\mathcal{J}, r, m) \models \varphi > \psi$ if
$(\mathcal{J}, r', m') \models \psi$ for all $(r', m') \in \texttt{closest}([\![\varphi]\!], (r, m), \ll)$

- $\texttt{closest}([\![\varphi]\!], (r, m), \ll)$ consists of the points closest to (r, m) w.r.t. $\ll$ satisfying φ.
- $\varphi > \psi$ is true at (r, m) if ψ is true at all the closest points to (r, m) where φ is true
  - This is just the Stalnaker/Lewis definition.

We are mainly interested in $do(i, a) > \psi$:

- Would ψ be true if i performed a?

Conditions on $\ll$

How should we define $\ll$?

- Intuitively, it depends on the protocol of interest
- We want minimize deviations from the protocol

$\mathcal{R}^+$ consists of all runs generated by all possible protocols.

- An *order generator* o maps protocols P to order $o(P) = \ll^P$ on the points in $\mathcal{R}^+$
- *o respects protocols* if, for all P, the closest points (w.r.t. $o(P) = \ll^P$) to (r, m) where i performs action a are points (r', m) such that
  - $r(m) = r'(m)$
    - up to time m, same thing happens in r and r'
  - i performs action a at (r', m)
  - otherwise, all agents follow protocol P
- Intuitively, $(r_1, m_1) \ll^P_{(r,m)} (r_2, m_2)$ if (r_1, m_1) if involves fewer deviations from P than (r_2, m_2).

Belief Formalized

Give semantics to belief using ideas of Spohn.

- A *ranking function* κ associates with every run either a natural number or ∞.
  - Bigger numbers mean the run is less likely
- $\mathsf{min}_i^\kappa(r,m) \;=\; \min\{\kappa(r') \mid r' \in \mathcal{R}^+, r_i'(m') = r_i(m)\}$
  - $\mathsf{min}_i^\kappa(r,m)$ is the smallest rank that i considers possible at (r,m)
- $(\mathcal{I}, \ll, \kappa, r, m) \models B_i\varphi$ iff $(\mathcal{I}, \ll, \kappa, r', m') \models \varphi$ for all (r', m') such that $\kappa(r') = \mathsf{min}_i^\kappa(r,m)$ and $r_i'(m') = r_i(m)$.
  - i believes φ is φ is true at all points that i considers possible that have minimal rank
  - This definition of belief satisfies KD45.

Conditions on κ

How should we define κ?

- Intuitively, it depends on the protocol of interest
- We want minimize deviations from the protocol

κ is *P-compatible* (in context γ) if runs of P are the "most likely" runs according to κ

- $\kappa(r) = 0$ iff $r \in \mathcal{R}(P, \gamma)$

Examples

1. $\kappa(r) = 0$ if $r \in \mathcal{R}(P, \gamma)$, else $\kappa(r) = 1$
2. $\kappa(r)$ counts number of deviations from a run of P

Lemma: If κ is P-compatible in γ and $(r, m) \in \mathcal{R}(P, \gamma)$, then $(\mathcal{R}(P, \gamma), r, m) \models K_i\varphi$ iff $(\mathcal{R}^+, \kappa, r, m) \models B_i\varphi$.

- With P-compatible rankings, belief in $\mathcal{R}^+$ acts like knowledge in $\mathcal{R}(P, \gamma)$.
- Can recover $\mathcal{R}(P, \gamma)$ from $\mathcal{R}^+$.

A *ranking generator* σ maps protocols P to rankings κ^P on the runs in $\mathcal{R}^+$.

- σ is *deviation compatible* (in context γ) if $\sigma(P)$ is P-compatible.

A Recap

We have a lot of machinery:

- Order generators that map protocols to orders on points in $\mathcal{R}^+$.
  - Needed to give semantics to counterfactuals.
- Ranking generators that map protocols to rankings on runs in $\mathcal{R}^+$.
  - Needed to give semantics to belief.
- Order generators that respect protocols and deviation compatible ranking generators give orders/rankings that are compatible with the underlying protocol.

Why bother?

- This is what we need to make sense out of *counterfactual belief-based (cbb) programs*:
  - programs with tests involving counterfactuals and belief

Counterfactual Belief-Based Programs

- An *extended context* (γ, o, σ) consists of a context γ, an order generator o and a ranking generator σ.
- Given a counterfactual belief-based program and $\mathcal{J} = (\mathcal{R}, \ll, \kappa, \pi)$, get a protocol $P = \mathsf{Pg}_{kb}^{\mathcal{J}}$ in context γ:
  - Use $\mathcal{J}$ to determine outcome of knowledge tests
  - Use $\ll$ to determine truth of counterfactuals
  - Use κ to determine beliefs
- If $\mathcal{J} = (\mathcal{R}(P, \gamma), o(P), \sigma(P))$, then P implements Pg_{kb} in extended context (γ, o, σ).

Key point: Cbb programs extend kb programs in extended contexts where σ is deviation compatible:

- If σ is deviation compatible, P implements a kb program Pg_{kb} in context γ iff P implements cbb program Pg_{kb}^{B} in extended context (γ, o, σ).

The Bit Transmission Problem Again

Consider three different contexts:

- γ_1: messages guaranteed to be within 5 rounds;
- γ_2: messages guaranteed to arrive eventually, but no upper bound on message delivery time;
- γ_3: messages guaranteed to arrive eventually, but only if sent infinitely often.

EC_i consists of all extended contexts of the form (γ_i, o, σ), where

- o respects protocols
- σ is deviation compatible

Consider two cbb protocols:

$\mathsf{BT}^{>}$: **if** $B_S(do(S, \mathsf{skip}) > \Diamond recbit)$ **then** skip **else** $\mathsf{sendbit}$
$\mathsf{BT}^{\Diamond B}$: **if** $B_S(do(S, \mathsf{skip}) > \Diamond B_R(bit))$ **then** skip **else** $\mathsf{sendbit}$

Theorem: Both $\mathsf{BT}^{>}$ and $\mathsf{BT}^{\Diamond B}$ solve the bit-transmission problem in all the extended contexts $EC_1 \cup EC_2 \cup EC_3$, and are implementable in each of these contexts.

EC_1

Recall: in EC_1, messages arrive within 5 rounds.

$P^1(k,m)$: **if** ($\mathit{time} = k$, $\mathit{bit} = 0$) or ($\mathit{time} = m$, $\mathit{bit} = 1$) **then** $\mathsf{sendbit}$ **else** skip.

Lemma:
(a) $P^1(k,m)$ implements $\mathsf{BT}^{>}$ in all contexts in EC_1
(b) $P^1(k,m)$ implements $\mathsf{BT}^{\Diamond B}$ in $\xi \in EC_1$ only if (i) $k = m$ and (ii) κ satisfies a technical property [see paper].

- Problem: $P^1(k,m)$ is sending an unnecessary message if messages are guaranteed to arrive.

$P^2(k,b)$: **if** $\mathit{time} = k$ and $\mathit{bit} = b$ **then** $\mathsf{sendbit}$ **else** skip.

- E.g., with $P^2(3,0)$, the sender sends 0 at round 3 if $\mathit{bit} = 0$, and nothing if $\mathit{bit} = 1$.

Lemma: Every instance of $P^2(k,b)$ implements $\mathsf{BT}^{\Diamond B}$ in every context in EC_1; no instance of $P^2(k,b)$ implements $\mathsf{BT}^{>}$ in contexts in EC_1.

EC_2

Recall: in EC_2 messages arrive eventually.

- Now messages must be sent for both bit values

Lemma: Every instance of $P^1(k,m)$ implements $\mathsf{BT}^{\Diamond B}$ and $\mathsf{BT}^{>}$ in every context in EC_2; no instance of $P^2(k,b)$ implements $\mathsf{BT}^{>}$ or $\mathsf{BT}^{\Diamond B}$ in contexts in EC_2.

EC_3

Recall: in EC_3, messages must be sent infinitely often to guarantee delivery.

- $P^1(k,m)$ or $P^2(k,b)$ do not $\mathsf{BT}^{\Diamond B}$ or $\mathsf{BT}^{>}$ implement in any context in EC_3.
  - One message clearly isn't enough!

For $I \subseteq I\!N$, define protocol $P(I)$:

$$\textbf{if } \textit{time} \in I \textbf{ then } \mathsf{sendbit} \textbf{ else } \mathsf{skip}$$

Lemma: For all choices of I, $P(I)$ does not implement $\mathsf{BT}^{>}$ or $\mathsf{BT}^{\Diamond B}$ in any context in EC_3.

- This is the *procrastinator's paradox*: can always postpone sending the message one more round.

P^ω: **if** $\textit{time} = 0$ or $\mathsf{sendbit}$ performed in previous round **then** $\mathsf{sendbit}$ **else** skip.

Lemma: P^ω implements both $\mathsf{BT}^{>}$ and $\mathsf{BT}^{\Diamond B}$ in every context in EC_3.

- But $P(I\!N)$ and P^ω generate the same system!
- What matters is what they do "off the beaten path".

Conclusions

We have presented a framework that allows counterfactual reasoning in protocols.

- Permits the design of efficient high-level protocols.
  - Current work: applying these ideas to *global funcion computation*:
    - computing function of values on a network
    - E.g.: leader election
- Approach may shed light on philosophical issues involved with counterfactuals
  - What worlds are "closest" depends on the protocol
- Approach may also shed light on equilibrium notions in game theory
  - You are in equilibrium if you would not be better off had you done otherwise.
    - But it also matters what you would have done "off the beaten path".