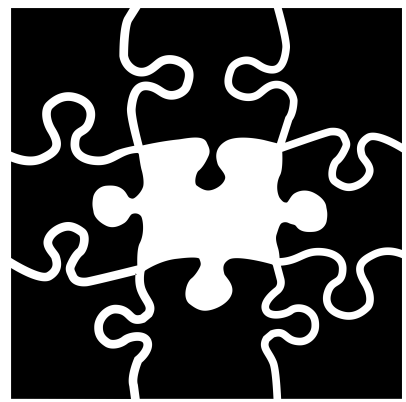
A Model Checker for the Hardest Logic Puzzle Ever

Malvin Gattinger – ILLC, Amsterdam



2016-05-10 – PhDs in Logic VIII – Darmstadt

The Hardest Logic Puzzle Ever

Three gods A, B, and C are called, in some order, True, False, and Random. True always speaks truly, False always speaks falsely, but whether Random speaks truly or falsely is a completely random matter.

Your task is to determine the identities of A, B, and C by asking three yes-no questions; each question must be put to exactly one god.

The gods understand English, but will answer all questions in their own language, in which the words for yes and no are 'da' and 'ja', in some order. You do not know which word means which.

This version is from (Boolos 1996), for older variants see R. Smullyan: "What is the name of this book." (1978)

What makes the HLPE hard?

- Who is who? $\Rightarrow 3! = 6$ possibilities
- What means what? $\Rightarrow 2$ possibilities

$\Rightarrow 12$ possibilities:

`ja TFR` `ja TRF` `ja FRT` `ja FTR` `ja RTF` `ja RFT`

`da TFR` `da TRF` `da FRT` `da FTR` `da RTF` `da RFT`

But we can only ask three questions $\Rightarrow$ distinguish $2^3 = 8$ cases!?

Oh, but we don't have to learn what means what!

`ja TFR`

`da TFR`

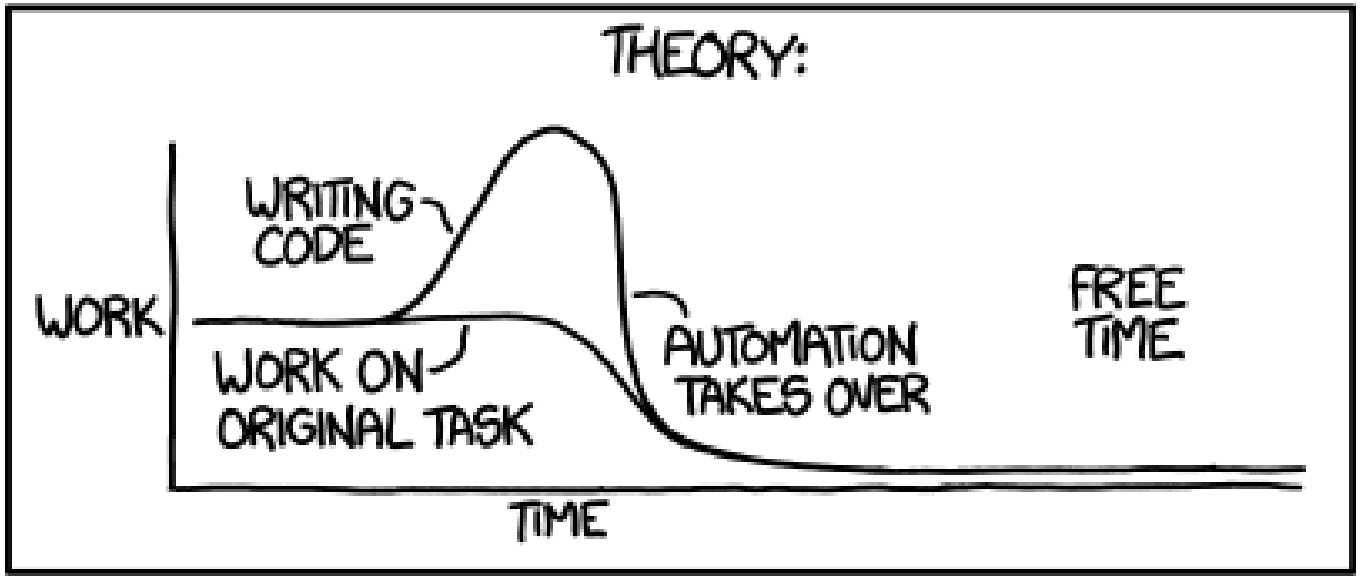
Spoiler Alert!

Philosopher (B. Rabern and Rabern 2008): Oh, that's easy, you first ask B whether his answer would be "ja" if you would ask them whether "A" is Random. If they say 'ja', then ask C if they would answer 'ja' if you would ask them whether they are True. If . . .


WHAT DO WE SAY TO THAT ASSIGNMENT?
NOT TODAY

Can't we just calculate whether a solution is correct?

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"



The Plan

1. Find a logic to express HLPE and solutions
2. Implement model checking
3. Profit

Logic to the rescue!

Question and Utterance logic with Agent Types
(Liu and Wang 2013)

A variant of Dynamic Epistemic Logic (DEL)

- epistemic: knowledge K_a using the modal logic S5
  "a knows that ϕ iff in all worlds a considers possible ϕ holds"
- dynamic: announcements $[!\phi]$ change the model
  "something is true after announcing ϕ iff it is true in $\mathcal{M}^{\phi}$"

with extras:

- Agency: $[!_a\phi]\phi$
- Questions: $[?_a\phi]$
- Utterances: $[!_a u]\phi$
- Agent Types: $\eta(a)$

Question and Utterance logic with Agent Types

Syntax:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid K_a\phi \mid \eta(a) \mid [!_a u]\phi \mid [?_a\phi]\phi \mid [!_a]\phi \mid$$

where a is an agent (A, B, C), η is a type (True, False, Random) and u is an utterance (ja, da).

Semantics: Given a model $\mathcal{M}$ with a valuation for propositions, types and utterances, and a current questions μ define

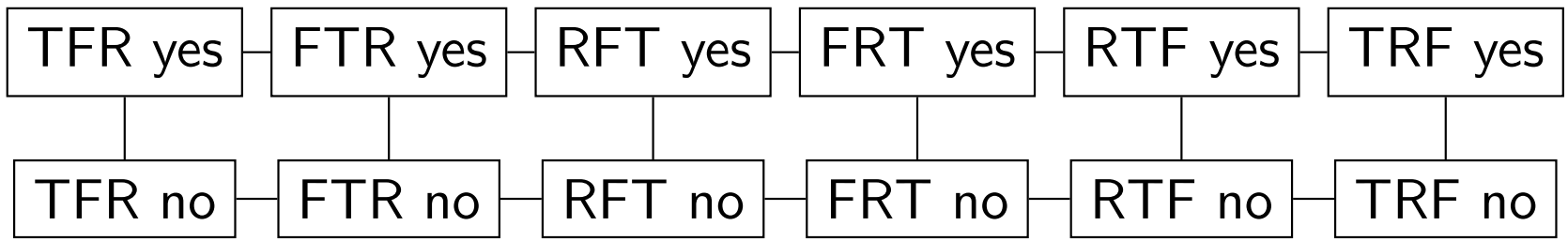
$$\mathcal{M}, s \models_\mu \phi \iff \ldots$$

Motivation: Model checking HLPE and solutions.

Implementation 1: HLPE in PQLTTU

What is a puzzle?

1. A model:

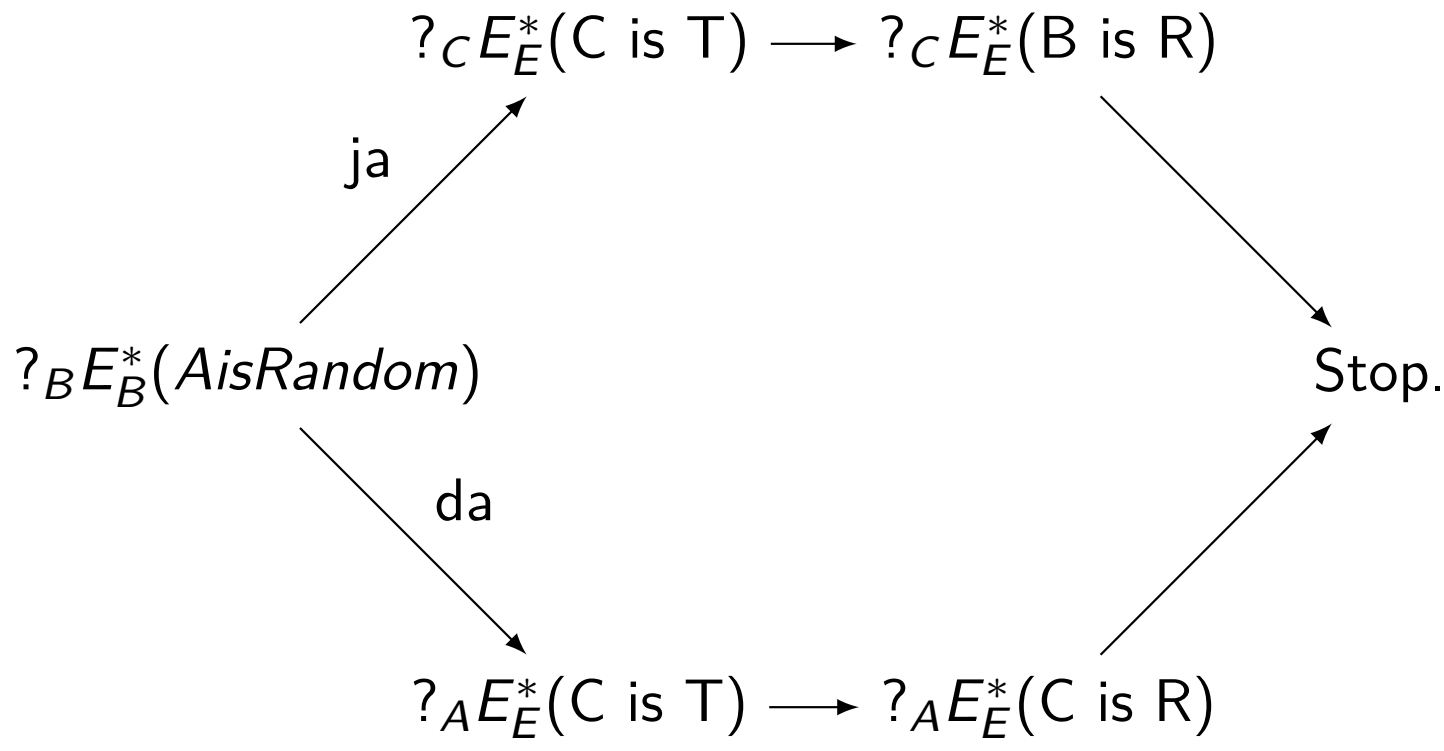


2. A goal: $K_i(TFR) \vee \cdots \vee K_i(TRF)$

Implementation 1: HLPE in PQLTTU

What is a possible solution? A *strategy*:

- Ask ...
- If answer is ... then ...



Implementation 1: HLPE in PQLTTU

What is a possible solution? A *strategy*:

- Ask ...
- If answer is ... then ...

Every strategy generates a set of *sequences*:

- Ask ϕ_1, get answer ψ_1^a, ask ϕ_2^a, get answer ψ_2^a ...
- Ask ϕ_1, get answer ψ_1^b, ask ϕ_2^b, get answer ψ_2^b ...
- ...

Each sequence gives us a formula saying that it succeeds:

$$[?_a\phi_1]\langle !_a\rangle\top \wedge [?_a\phi_1][!_a\psi_1^a]\ldots[?_a\phi_n^a][!_a\psi_n]\theta$$

$$[?_a\phi_1]\langle !_a\rangle\top \wedge [?_a\phi_1][!_a\psi_2^b]\ldots[?_a\phi_n^b][!_a\psi_n]\theta$$

$$\vdots$$

From Puzzle Solving to Model Checking

$$\text{Strategy} \rightsquigarrow \text{Sequences} \rightsquigarrow \text{Formulas}$$

Change the question from: "Does this strategy solve the puzzle?"
to "Are theses formulas valid on the HLPE model?"

Implementation 2: PQLTTU in Haskell

Formulas:

```
data QUForm =
  Top | Prp Prop | AgentName `IsA` AgentType
  | Neg QUForm | Conj [QUForm] | Disj [QUForm]
  | Impl QUForm QUForm
  | K AgentName QUForm
  | C [AgentName] QUForm
  | Say AgentName Utterance QUForm
  | Ask AgentName QUForm QUForm
  | Ans AgentName QUForm
  | DmSay AgentName Utterance QUForm
  | DmAsk AgentName QUForm QUForm
  | DmAns AgentName QUForm
```

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid K_a\phi \mid \eta(a) \mid [!_a u]\phi \mid [?_a\phi]\phi \mid [!_a]\phi \mid$$

Implementation 2: PQLTTU in Haskell

Utterances, Meanings, Models:

```
type Utterance = String

data Meaning = Mng String (QUForm -> QUForm)

data Model = Mo
  [(World, (Label, TypeLabel, UtteranceLabel))]
  [(AgentName, Partition World)]

type Scene = (Model, World)

type Context = Maybe (AgentName,QUForm)
```

Implementation 2: PQLTTU in Haskell

Semantics I:

```
qIsTrue :: Scene -> Context -> QUForm -> Bool

qIsTrue _  _ Top           = True
qIsTrue sc _ (Prp p)       = p `elem` labelAt sc
qIsTrue sc _ (a `IsA` t)   = typeAtOf sc a == t
qIsTrue sc c (Neg f)       = not (qIsTrue sc c f)
qIsTrue sc c (Conj fs)     = all (qIsTrue sc c) fs
qIsTrue sc c (Disj fs)     = any (qIsTrue sc c) fs
qIsTrue sc c (Impl f g)    = not (qIsTrue sc c f)
                               || qIsTrue sc c g

qIsTrue sc _ (Ask a f g) = qIsTrue sc (Just (a,f)) g
```

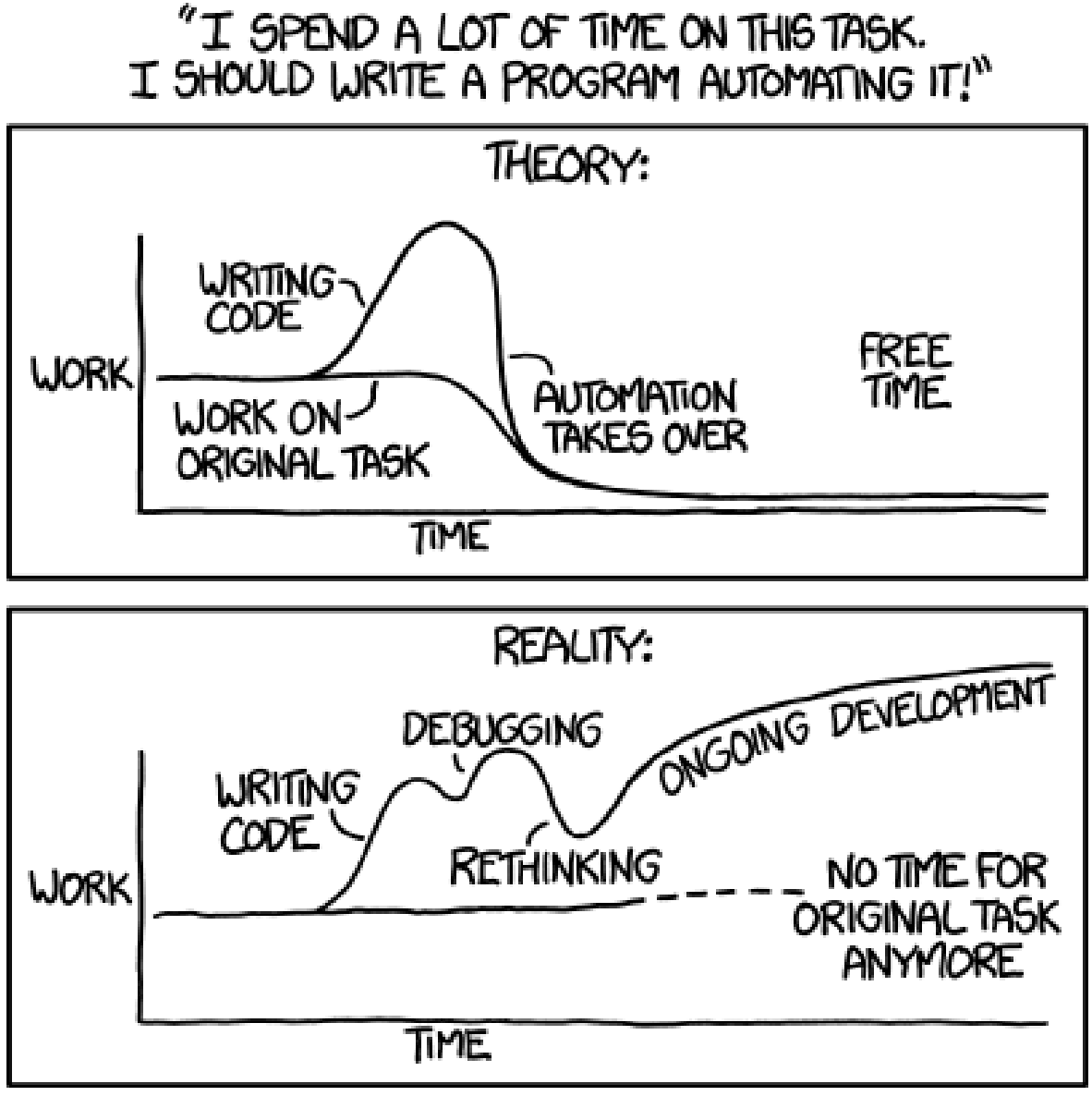
Implementation 2: PQLTTU in Haskell

Semantics II:

```
qIsTrue sc@(m,w) c@(Just (a1,q)) (Say a2 u phi) =
  (a1 == a2) && f `elem` [q,Neg q] && qCanSay sc a1 f
    `implies` qIsTrue (qAnnounce m c a1 u, w) Nothing phi
  where
    f = meanfctAtOfGiven sc u q

qAnnounce :: Model -> Context -> AgentName -> Utterance -> Model
qAnnounce oldm@(Mo oldws oldpss) c a u = Mo ws pss where
  ws = filter check oldws
  check (v,_) = qIsTrue (oldm,v) Nothing (agtf a f) where
    AgT _ agtf = typeAtOf (oldm,v) a
    f = meanfctAtOfGiven (oldm,v) u q
    Just (_,q) = c
  pss = map (fmap strip) oldpss
  strip set =
    filter ([]/=) map (filter (`elem` map fst ws)) set
```

Only one scary Haskell slide left!



https://xkcd.com/1319

Implementation 2: PQLTTU in Haskell

Validity and Puzzles:

```
qIsValid m@(Mo ws _) f =
  and [ qIsTrue (m,w) Nothing f | w <- map fst ws ]

type Puzzle   = (Model,QUForm)
type Question = (AgentName,QUForm)
data Strategy =
  FState
  | QState Question [(Utterance,Strategy)]

solves :: Strategy -> Puzzle -> Bool
solves strategy (model,goal) = all
  (qIsValid model . seq2f goal)
  (sequencesOf strategy)
```

The HLPE Model

```
1:  [], A:TT B:LL C:LT D:TT, ja=yes da=no
2:  [], A:LL B:TT C:LT D:TT, ja=yes da=no
3:  [], A:LT B:LL C:TT D:TT, ja=yes da=no
4:  [], A:LL B:LT C:TT D:TT, ja=yes da=no
5:  [], A:LT B:TT C:LL D:TT, ja=yes da=no
6:  [], A:TT B:LT C:LL D:TT, ja=yes da=no
7:  [], A:TT B:LL C:LT D:TT, ja=no  da=yes
8:  [], A:LL B:TT C:LT D:TT, ja=no  da=yes
9:  [], A:LT B:LL C:TT D:TT, ja=no  da=yes
10: [], A:LL B:LT C:TT D:TT, ja=no  da=yes
11: [], A:LT B:TT C:LL D:TT, ja=no  da=yes
12: [], A:TT B:LT C:LL D:TT, ja=no  da=yes
A: [[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]]
B: [[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]]
C: [[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]]
D: [[1,2,3,4,5,6,7,8,9,10,11,12]]
```

Solving HLPE in a second

```
GHCi, version 7.10.3: http://www.haskell.org/ghc/
...
Ok, modules loaded: QULogic, HELP, AGTYPES.

*QULogic> rabern
Ask B if >>[B?(A is a LT)]<B: ja>T<<.
  If they say 'ja', then:
    Ask C if >>[C?(C is a TT)]<C: ja>T<<.
      No matter what they say, :
        Ask C if >>[C?(B is a LT)]<C: ja>T<<.
          No matter what they say, stop.
  If they say 'da', then:
    Ask A if >>[A?(A is a TT)]<A: ja>T<<.
      No matter what they say, :
        Ask A if >>[A?(B is a LT)]<A: ja>T<<.
          No matter what they say, stop.

*QULogic> rabern `solves` hlpe
True
(0.07 secs, 13,475,456 bytes)
```

Future Work

- bounded search
  - find solutions automatically
  - verify unsolvability
- implement translations to PAL
  - performance?
- automatically generate puzzles and solutions
- more agent types
  - "... giraffe always answers to the previous question ..."

Source Code and References

`https://github.com/m4lvin/mchlpe`

Boolos, George. 1996. "The Hardest Logic Puzzle Ever." *The Harvard Review of Philosophy* 6 (1): 62–65.

Liu, Fenrong, and Yanjing Wang. 2013. "Reasoning About Agent Types and the Hardest Logic Puzzle Ever." *Minds and Machines* 23 (1). Springer: 123–61. doi:10.1007/s11023-012-9287-x.

Rabern, Brian, and Landon Rabern. 2008. "A Simple Solution to the Hardest Logic Puzzle Ever." *Analysis* 68 (2). Oxford University Press: 105–12. doi:10.1093/analys/68.2.105.