Solution to the xkcd Blue Eyes puzzle

Maya Kdzioka

March 17, 2025

Abstract

In a puzzle published by Randall Munroe [2], perfect logicians forbidden from communicating are stranded on an island, and may only leave once they have figured out their own eye color. We present a method of modeling the behavior of perfect logicians and formalize a solution of the puzzle.

Contents

1 Introduction

The original problem statement [2] explains the puzzle well:

> A group of people with assorted eye colors live on an island. They are all perfect logicians – if a conclusion can be logically deduced, they will do it instantly. No one knows the color of their eyes. Every night at midnight, a ferry stops at the island. Any islanders who have figured out the color of their own eyes then leave the island, and the rest stay. Everyone can see everyone else at all times and keeps a count of the number of people they see with each eye color (excluding themselves), but they cannot otherwise communicate. Everyone on the island knows all the rules in this paragraph.
>
> On this island there are 100 blue-eyed people, 100 brown-eyed people, and the Guru (she happens to have green eyes). So any given blue-eyed person can see 100 people with brown eyes and 99 people with blue eyes (and one with green), but that does not tell him his own eye color; as far as he knows the totals could be 101 brown and 99 blue. Or 100 brown, 99 blue, and he could have red eyes.
>
> The Guru is allowed to speak once (let's say at noon), on one day in all their endless years on the island. Standing before the islanders, she says the following:
>
> "I can see someone who has blue eyes."
>
> Who leaves the island, and on what night?

It might seem weird that the Guru's declaration gives anyone any new information. For an informal discussion, see [1, Section 1.1].

2 Modeling the world

We begin by fixing two type variables: $'color$ and $'person$. The puzzle doesn't specify how many eye colors are possible, but four are mentioned. Crucially, we must assume they are distinct. We specify the existence of colors other than blue and brown, even though we don't mention them later, because when blue and brown are the only possible colors, the puzzle has a different solution — the brown-eyed logicians may leave one day after the blue-eyed ones.

We refrain from specifying the exact population of the island, choosing to only assume it is finite and denote a specific person as the Guru.

We could also model the Guru as an outside entity instead of a participant. This doesn't change the answer and results in a slightly simpler proof, but is less faithful to the problem statement.

context
  fixes *blue brown green red* :: *'color*
  assumes *colors-distinct*: *distinct* [*blue*, *brown*, *green*, *red*]

  fixes *guru* :: *'person*
  assumes *finite* (*UNIV* :: *'person set*)
begin

It's slightly tricky to formalize the behavior of perfect logicians. The representation we use is centered around the type of a *world*, which describes the entire state of the environment. In our case, it's a function $'person \Rightarrow 'color$ that assigns an eye color to everyone.[1]

The only condition known to everyone and not dependent on the observer is Guru's declaration:

definition *valid* :: ($'person \Rightarrow 'color$) $\Rightarrow$ *bool* **where**
  $valid\ w \longleftrightarrow (\exists p.\ p \neq guru \wedge w\ p = blue)$

We then define the function $possible\ n\ p\ w\ w'$, which returns *True* if on day n the potential world w' is plausible from the perspective of person p, based on the observations they made in the actual world w.

Then, $leaves\ n\ p\ w$ is *True* if p is able to unambiguously deduce the color of their own eyes, i.e. if it is the same in all possible worlds. Note that if p actually left many moons ago, this function still returns *True*.

fun *leaves* :: $nat \Rightarrow 'person \Rightarrow ('person \Rightarrow 'color) \Rightarrow bool$
  and *possible* :: $nat \Rightarrow 'person \Rightarrow ('person \Rightarrow 'color) \Rightarrow ('person \Rightarrow 'color) \Rightarrow bool$
  where
    $leaves\ n\ p\ w = (\forall w'.\ possible\ n\ p\ w\ w' \longrightarrow w'\ p = w\ p)$ |
    $possible\ n\ p\ w\ w' \longleftrightarrow valid\ w \wedge valid\ w'$
    $\wedge (\forall p' \neq p.\ w\ p' = w'\ p')$
    $\wedge (\forall n' < n.\ \forall p'.\ leaves\ n'\ p'\ w = leaves\ n'\ p'\ w')$

Naturally, the act of someone leaving can be observed by others, thus the two definitions are mutually recursive. As such, we need to instruct the simplifier to not unfold these definitions endlessly.

declare *possible.simps*[*simp del*] *leaves.simps*[*simp del*]

A world is possible if

1. The Guru's declaration holds.
2. The eye color of everyone but the observer matches.
3. The same people left on each of the previous days.

[1] We would introduce a type synonym, but at the time of writing Isabelle doesn't support including type variables fixed by a locale in a type synonym.

Moreover, we require that the actual world w is *valid*, so that the relation is symmetric:

```
lemma possible-sym: possible n p w w' = possible n p w' w
  by (auto simp: possible.simps)
```

In fact, *possible n p* is an equivalence relation:

```
lemma possible-refl: valid w ⟹ possible n p w w
  by (auto simp: possible.simps)

lemma possible-trans: possible n p w1 w2 ⟹ possible n p w2 w3 ⟹ possible n p w1 w3
  by (auto simp: possible.simps)
```

3 Eye colors other than blue

Since there is no way to distinguish between the colors other than blue, only the blue-eyed people will ever leave. To formalize this notion, we define a function that takes a world and replaces the eye color of a specified person. The original color is specified too, so that the transformation composes nicely with the recursive hypothetical worlds of *local.possible*.

```
definition try-swap :: 'person ⇒ 'color ⇒ 'color ⇒ ('person ⇒ 'color) ⇒ ('person ⇒ 'color) where
  try-swap p c_1 c_2 w x = (if c_1 = blue ∨ c_2 = blue ∨ x ≠ p then w x else transpose c_1 c_2 (w x))

lemma try-swap-valid[simp]: valid (try-swap p c_1 c_2 w) = valid w
  by (cases ‹c_1 = blue›; cases ‹c_2 = blue›)
    (auto simp add: try-swap-def valid-def transpose-eq-iff)

lemma try-swap-eq[simp]: try-swap p c_1 c_2 w x = try-swap p c_1 c_2 w' x ⟷ w x = w' x
  by (auto simp add: try-swap-def transpose-eq-iff)

lemma try-swap-inv[simp]: try-swap p c_1 c_2 (try-swap p c_1 c_2 w) = w
  by (rule ext) (auto simp add: try-swap-def swap-id-eq)

lemma leaves-try-swap[simp]:
  assumes valid w
  shows leaves n p (try-swap p' c_1 c_2 w) = leaves n p w
  using assms
proof (induction n arbitrary: p w rule: less-induct)
  case (less n)
  have leaves n p w if leaves n p (try-swap p' c_1 c_2 w) for w
  proof (unfold leaves.simps; rule+)
    fix w'
    assume possible n p w w'
    then have possible n p (try-swap p' c_1 c_2 w) (try-swap p' c_1 c_2 w')
      by (fastforce simp: possible.simps less.IH)
    with ‹leaves n p (try-swap p' c_1 c_2 w)› have try-swap p' c_1 c_2 w' p = try-swap p' c_1 c_2 w p
      unfolding leaves.simps
      by simp
    thus w' p = w p by simp
  qed

  with try-swap-inv show ?case by auto
qed
```

This lets us prove that only blue-eyed people will ever leave the island.

```
proposition only-blue-eyes-leave:
  assumes leaves n p w and valid w
  shows w p = blue
proof (rule ccontr)
```

```
  assume w p ≠ blue
  then obtain c where c: w p ≠ c  c ≠ blue
    using colors-distinct
    by (metis distinct-length-2-or-more)

  let ?w' = try-swap p (w p) c w
  have possible n p w ?w'
    using ‹valid w› apply (simp add: possible.simps)
    by (auto simp: try-swap-def)
  moreover have ?w' p ≠ w p
    using c ‹w p ≠ blue› by (auto simp: try-swap-def)
  ultimately have ¬ leaves n p w
    by (auto simp: leaves.simps)
  with assms show False by simp
qed
```

4 The blue-eyed logicians

We will now consider the behavior of the logicians with blue eyes. First, some simple lemmas. Reasoning about set cardinalities often requires considering infinite sets separately. Usefully, all sets of people are finite by assumption.

```
lemma people-finite[simp]: finite (S::'person set)
proof (rule finite-subset)
  show S ⊆ UNIV by auto
  show finite (UNIV::'person set) by fact
qed
```

Secondly, we prove a destruction rule for *local.possible*. It is strictly weaker than the definition, but thanks to the simpler form, it's easier to guide the automation with it.

```
lemma possibleD-colors:
  assumes possible n p w w' and p' ≠ p
  shows w' p' = w p'
  using assms unfolding possible.simps by simp
```

A central concept in the reasoning is the set of blue-eyed people someone can see.

```
definition blues-seen :: ('person ⇒ 'color) ⇒ 'person ⇒ 'person set where
  blues-seen w p = {p'. w p' = blue} − {p}

lemma blues-seen-others:
  assumes w p' = blue and p ≠ p'
  shows w p = blue ⟹ card (blues-seen w p) = card (blues-seen w p')
    and w p ≠ blue ⟹ card (blues-seen w p) = Suc (card (blues-seen w p'))
proof −
  assume w p = blue
  then have blues-seen w p' = blues-seen w p ∪ {p} − {p'}
    by (auto simp add: blues-seen-def)
  moreover have p ∉ blues-seen w p
    unfolding blues-seen-def by auto
  moreover have p' ∈ blues-seen w p ∪ {p}
    unfolding blues-seen-def using ‹p ≠ p'› ‹w p' = blue› by auto
  ultimately show card (blues-seen w p) = card (blues-seen w p')
    by simp
next
  assume w p ≠ blue
  then have blues-seen w p' = blues-seen w p − {p'}
    by (auto simp add: blues-seen-def)
```

moreover have $p' \in$ *blues-seen w p*
unfolding *blues-seen-def* **using** ‹$p \neq p'$› ‹$w\ p' = blue$› **by** *auto*
ultimately show *card* (*blues-seen w p*) = *Suc* (*card* (*blues-seen w p'*))
by (*simp only*: *card-Suc-Diff1 people-finite*)
qed

lemma *blues-seen-same*[*simp*]:
assumes *possible n p w w'*
shows *blues-seen w' p* = *blues-seen w p*
using *assms*
by (*auto simp*: *blues-seen-def possible.simps*)

lemma *possible-blues-seen*:
assumes *possible n p w w'*
assumes $w\ p' = blue$ **and** $p \neq p'$
shows $w'\ p = blue \Longrightarrow$ *card* (*blues-seen w p*) = *card* (*blues-seen w' p'*)
and $w'\ p \neq blue \Longrightarrow$ *card* (*blues-seen w p*) = *Suc* (*card* (*blues-seen w' p'*))
using *possibleD-colors*[*OF* ‹*possible n p w w'*›] **and** *blues-seen-others assms*
by (*auto simp flip*: *blues-seen-same*)

Finally, the crux of the solution. We proceed by strong induction.

lemma *blue-leaves*:
assumes $w\ p = blue$ **and** *valid w*
and *guru*: $w\ guru \neq blue$
shows *leaves n p w* $\longleftrightarrow n \geq$ *card* (*blues-seen w p*)
using *assms*
proof (*induction n arbitrary*: *p w rule*: *less-induct*)
case (*less n*)
show *?case*
proof
— First, we show that day n is sufficient to deduce that the eyes are blue.
assume $n \geq$ *card* (*blues-seen w p*)
have $w'\ p = blue$ **if** *possible n p w w'* **for** *w'*
proof (*cases card* (*blues-seen w' p*))
case *0*
moreover from ‹*possible n p w w'*› **have** *valid w'*
by (*simp add*: *possible.simps*)
ultimately show $w'\ p = blue$
unfolding *valid-def blues-seen-def* **by** *auto*
next
case (*Suc k*)
— We consider the behavior of somebody else, who also has blue eyes.
then have *blues-seen w' p* $\neq \{\}$
by *auto*
then obtain p' **where** $w'\ p' = blue$ **and** $p \neq p'$
unfolding *blues-seen-def* **by** *auto*
then have $w\ p' = blue$
using *possibleD-colors*[*OF* ‹*possible n p w w'*›] **by** *simp*

have $p \neq guru$
using ‹$w\ p = blue$› **and** ‹$w\ guru \neq blue$› **by** *auto*
hence $w'\ guru \neq blue$
using ‹$w\ guru \neq blue$› **and** *possibleD-colors*[*OF* ‹*possible n p w w'*›] **by** *simp*

have *valid w'*
using ‹*possible n p w w'*› **unfolding** *possible.simps* **by** *simp*

show $w'\ p = blue$

```
      proof (rule ccontr)
        assume w' p ≠ blue
        — If our eyes weren't blue, then p' would see one blue-eyed person less than us.
        with possible-blues-seen[OF ‹possible n p w w'› ‹w p' = blue› ‹p ≠ p'›]
        have *: card (blues-seen w p) = Suc (card (blues-seen w' p'))
          by simp
        — By induction, they would've left on day k = blues-seen w' p'.
        let ?k = card (blues-seen w' p')
        have ?k < n
          using ‹n ≥ card (blues-seen w p)› and * by simp
        hence leaves ?k p' w'
          using ‹valid w'› ‹w' p' = blue› ‹w' guru ≠ blue›
          by (intro less.IH[THEN iffD2]; auto)
        — However, we know that actually, p' didn't leave that day yet.
        moreover have ¬ leaves ?k p' w
        proof
          assume leaves ?k p' w
          then have ?k ≥ card (blues-seen w p')
            using ‹?k < n› ‹w p' = blue› ‹valid w› ‹w guru ≠ blue›
            by (intro less.IH[THEN iffD1]; auto)

          have card (blues-seen w p) = card (blues-seen w p')
            by (intro blues-seen-others; fact)
          with * have ?k < card (blues-seen w p')
            by simp
          with ‹?k ≥ card (blues-seen w p')› show False by simp
        qed
        moreover have leaves ?k p' w' = leaves ?k p' w
          using ‹possible n p w w'› ‹?k < n›
          unfolding possible.simps by simp
        ultimately show False by simp
      qed
    qed
    thus leaves n p w
      unfolding leaves.simps using ‹w p = blue› by simp
  next
    — Then, we show that it's not possible to deduce the eye color any earlier.
    {
      assume n < card (blues-seen w p)
      — Consider a hypothetical world where p has brown eyes instead. We will prove that this world is
possible.
      let ?w' = w(p := brown)
      have ?w' guru ≠ blue
        using ‹w guru ≠ blue› ‹w p = blue›
        by auto
      have valid ?w'
      proof −
        from ‹n < card (blues-seen w p)› have card (blues-seen w p) ≠ 0 by auto
        hence blues-seen w p ≠ {}
          by auto
        then obtain p' where p' ∈ blues-seen w p
          by auto
        hence p ≠ p' and w p' = blue
          by (auto simp: blues-seen-def)
        hence ?w' p' = blue by auto
        with ‹?w' guru ≠ blue› show valid ?w'
          unfolding valid-def by auto
      qed
```

```
    moreover have leaves n′ p′ w = leaves n′ p′ ?w′ if n′ < n for n′ p′
    proof −
      have not-leavesI: ¬leaves n′ p′ w′
        if valid w′  w′ guru ≠ blue and P: w′ p′ = blue ⟹ n′ < card (blues-seen w′ p′) for w′
      proof (cases w′ p′ = blue)
        case True
        then have leaves n′ p′ w′ ⟷ n′ ≥ card (blues-seen w′ p′)
          using less.IH ‹n′ < n› ‹valid w′› ‹w′ guru ≠ blue›
          by simp
        with P[OF ‹w′ p′ = blue›] show ¬leaves n′ p′ w′ by simp
      next
        case False
        then show ¬ leaves n′ p′ w′
          using only-blue-eyes-leave ‹valid w′› by auto
      qed

      have ¬leaves n′ p′ w
      proof (intro not-leavesI)
        assume w p′ = blue
        with ‹w p = blue› have card (blues-seen w p) = card (blues-seen w p′)
          apply (cases p = p′, simp)
          by (intro blues-seen-others; auto)
        with ‹n′ < n› and ‹n < card (blues-seen w p)› show n′ < card (blues-seen w p′)
          by simp
      qed fact+

      moreover have ¬ leaves n′ p′ ?w′
      proof (intro not-leavesI)
        assume ?w′ p′ = blue
        with colors-distinct have p ≠ p′ and ?w′ p ≠ blue by auto
        hence card (blues-seen ?w′ p) = Suc (card (blues-seen ?w′ p′))
          using ‹?w′ p′ = blue›
          by (intro blues-seen-others; auto)
        moreover have blues-seen w p = blues-seen ?w′ p
          unfolding blues-seen-def by auto
        ultimately show n′ < card (blues-seen ?w′ p′)
          using ‹n′ < n› and ‹n < card (blues-seen w p)›
          by auto
      qed fact+

      ultimately show leaves n′ p′ w = leaves n′ p′ ?w′ by simp
    qed
    ultimately have possible n p w ?w′
      using ‹valid w›
      by (auto simp: possible.simps)
    moreover have ?w′ p ≠ blue
      using colors-distinct by auto
    ultimately have ¬ leaves n p w
      unfolding leaves.simps
      using ‹w p = blue› by blast
  }
  then show leaves n p w ⟹ n ≥ card (blues-seen w p)
    by fastforce
 qed
qed
```

This can be combined into a theorem that describes the behavior of the logicians based on the objective count of blue-eyed people, and not the count by a specific person. The xkcd puzzle is the instance where $n = 99$.

```
theorem blue-eyes:
  assumes card {p. w p = blue} = Suc n and valid w and w guru ≠ blue
  shows leaves k p w ⟷ w p = blue ∧ k ≥ n
proof (cases w p = blue)
  case True
  with assms have card (blues-seen w p) = n
    unfolding blues-seen-def by simp
  then show ?thesis
    using ‹w p = blue› ‹valid w› ‹w guru ≠ blue› blue-leaves
    by simp
next
  case False
  then show ?thesis
    using only-blue-eyes-leave ‹valid w› by auto
qed

end
```

5 Future work

After completing this formalization, I have been made aware of epistemic logic. The *possible worlds* model in section 2 turns out to be quite similar to the usual semantics of this logic. It might be interesting to solve this puzzle within the axiom system of epistemic logic, without explicit reasoning about possible worlds.

References

[1] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[2] Randall Munroe. Blue eyes — a logic puzzle. URL: https://xkcd.com/blue_eyes.html.