Games With Oracles That Lie

Joint work with Richard J. Lipton, Georgia Tech

Kenneth W. Regan
University at Buffalo (SUNY)

AMS Regional Meeting, 16 Sept., 2015

A Classic "Liar Puzzle"

At a fork in the road stands a man in armor. He is either a **Knight**, who always tells the truth, or a **Knave**, who always lies. One road leads to Wonderland, the other to Death. You may ask the man one question before choosing, and once you start down the Left or Right road you can't turn back. Can you survive?

Cf. Raymond Smullyan (1919–2017), *What is the Name of This Book?* [Showed GLL pages with Smullyan.]

Spoiler Alert. Answer (one of several variations on the theme):

Ask: "If I were to ask you whether the Left path leads to Wonderland, would you say *yes*?"

Solution Analysis

"If I asked you whether Left leads to W, would you say *yes*?"

1. First suppose the man is a Knight:
   - If he says 'yes' then it is true that if you asked him he would say 'yes' which would be true, so Left goes to W.
   - If he says 'no' then he truthfully would say no if you asked directly, so Left goes to D and Right goes to W.
2. Now suppose the man is a Knave:
   - If he says 'yes' then it is a lie, so if you asked him directly he really would say 'no'. But since he is a Knave, the 'no' to the direct question would be a lie, so Left really does go to W.
   - If he says 'no' then it is a lie, so if you asked directly he really would say 'yes'—but that would be a lie so Left really goes to D.

So regardless of whether the answer is a lie, if you hear 'yes' you should go Left, else go Right.

New Wrinkle: Knights Can Lie Once

Now suppose the man either is a Knave who always lies or a somewhat fallen K who can lie once. What then?

Can we do it in one question? Let's try changing the question slighlty:

Try: "If I were to ask you whether the Left path leads to Wonderland, **could** you say *yes*?"

If he is a Knave we have the same analysis. So suppose he is a Knight.

- If he says 'yes' then either he is using his one allowed lie or telling the truth and then could lie again.
    - If lying, then it is false that he could say ' yes.' So he could not say 'yes,' so he *would* say 'no,' and since he had used up his lie, it would be a truthful 'no,' so 'yes' means Left goes to W as before.
    - If telling the truth, then it is true that he could say 'yes.' But the latter could be his allowed lie. . .

Oracles That Can Lie $k = 1$ Times

"If I asked you whether Left leads to W, **could** you say 'yes'?"

- The first issue is that if only Right goes to W, the knight could truthfully say 'yes' to your question, because he *could* say 'yes' again. **So you don't know.**
- If he says 'no' as a K, then if he is lying, it must mean he could say 'yes' but since he would have used up his one allowed lie, the latter 'yes' would be true.
- So in this case, an initial 'no' includes *Left* going to W.
- This second issue definitely ruins things. So no go.
- You can modify the question to say, "If I were to ask you twice. . . " But that is kind-of cheating. Let's relax something else. . .

The New Setting

1. *No Indirect Questions*—must ask oracle which move is best.
2. *Allow Hypothetical Plays*—you may backtrack (until you *commit* to a choice in a path) but we count your steps as cost.
3. *Oracle is a K*—no knaves—but we may have more than one of them.
4. *Multiple Forking Paths*—which may come together again.
5. *Dragons*: At every other fork, a dragon may swoop down and bar all but one path with fire. Crazier? No, more familiar:

- Describes *Chess* or any two-person game with alternating turns.
- Dragon = opponent.
- W = Win for you, D = Draw or Loss (or any inferior result).
- Final positions are either W or D.
- Non-final positions are Winning for you—or for Dragon.
- Chess is **Hard** to play. How hard?

Chess and Complexity and Oracles

- Chess, Go, other games extend to $n \times n$ boards—with plays limited to length $h = n^{O(1)}$ by some "generalized 50-move rule."
- Given a position p, is it Winning?
- This problem is NP-**Hard**, indeed PSPACE-**Complete**.
- But computers are awfully good at the concrete forms of these games: chess *engines* **Deep Blue, Stockfish, Komodo. . .** ; **AlphaGo. . .**
- Hence they can be *oracles* to help us play.

Can *we* play perfectly even if the oracle has some *faults*?

- A non-faulty chess oracle K makes $\mathsf{P}^K = \mathsf{NP}^K = \mathsf{PSPACE}$.
- Question is "really about" **Fault-Tolerance** in computing.

The Basic Idea for a One-Error Oracle

- Starting with an initial position P with legal moves $m_1, \ldots, m_\ell$ and the premise—or promise—that P is W, focus on playing an optimal move from P.
- $k = 1$ means K can be faulty in at most one position in the game "tree" from P.
- Ask K to suggest a move m_i. Let Q be the new position.
- It's Dragon's turn, but you're in "Hypo-Play" mode: Dragon does not swoop. You can be the hypothetical Dragon.
- Ask K: What move should Dragon play at Q?
- If K lied at P, then K must find winning moves for Dragon here and later—and you lose the hypo-play.
- Hence if you *win* the hypo-play, you know m_i was not a lie and can *commit* to it.

Basic Idea—Continued

- If K lied in P, no use asking K to suggest another move m_j—it is faulty at P.
- But you can hypo-try m_j and ask whether the new position R_j is W for you.
    - There must be such a move, since P was W.
    - K canot lie about R_j since K used up one lie.
    - When K says some R_j is W, you can commit to the move m_j.
- So if you *lose* the hypo-play, *backtrack* to find the position P' along it where K lied.
- If $P' \neq P$, commit to m_i. Else $P' = P$ so find good m_j as above.
- Whew.
- When the basic idea for the $k = 1$ case won't fit on one Beamer frame, you know the whole thing is complicated...

Higher k, More Oracles—Complicated!

Theorem

Given an oracle K that makes at most k errors below P of height h, we can build an "engine" E_k that plays perfectly from P. The time however is exponential in k and h.

- Suppose instead we have *two* oracles K_1, K_2, collectively faulty in at most k positions.
- Idea: We can play them off against each other.
- Recursion then avoids excessive backtracking.
- Time becomes $O(h^k)$: still exponential in k but "better" in h.
- Further work on algorithms to improve these bounds is in progress... END for now, thanks!