The Story Next To the Story Behind the Results

"Pip"
Buff Georgia Unitech

Oxford University, 4 November 2015

My Association With Richard Lipton

- 1986 – Logic and P=?NP.

My Association With Richard Lipton

- 1986 – Logic and P=?NP.
- 1994–1997 – Keys and Codes and Graph Separators

My Association With Richard Lipton

- 1986 – Logic and P=?NP.
- 1994–1997 – Keys and Codes and Graph Separators
- 2008 – 60th Birthday Workshop

My Association With Richard Lipton

- 1986 – Logic and P=?NP.
- 1994–1997 – Keys and Codes and Graph Separators
- 2008 – 60th Birthday Workshop
- 2009–present – Working Up on the GLL Blog.

My Association With Richard Lipton

- 1986 – Logic and P=?NP.
- 1994–1997 – Keys and Codes and Graph Separators
- 2008 – 60th Birthday Workshop
- 2009–present – Working Up on the GLL Blog.
- Shared values of sharing values...

My Association With Richard Lipton

- 1986 – Logic and P=?NP.
- 1994–1997 – Keys and Codes and Graph Separators
- 2008 – 60th Birthday Workshop
- 2009–present – Working Up on the GLL Blog.
- Shared values of sharing values...and Ideas.

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.
- Can usefully abstract as "optimal" and "sub-optimal" values; optimal 1.0.

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.
- Can usefully abstract as "optimal" and "sub-optimal" values; optimal 1.0.
- Starting node P is winning (gives optimal value) for player to move.

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.
- Can usefully abstract as "optimal" and "sub-optimal" values; optimal 1.0.
- Starting node P is winning (gives optimal value) for player to move.
- Limit is K errors...

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.
- Can usefully abstract as "optimal" and "sub-optimal" values; optimal 1.0.
- Starting node P is winning (gives optimal value) for player to move.
- Limit is K errors...

1. below P, or

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.
- Can usefully abstract as "optimal" and "sub-optimal" values; optimal 1.0.
- Starting node P is winning (gives optimal value) for player to move.
- Limit is K errors...

1. below P, or

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.
- Can usefully abstract as "optimal" and "sub-optimal" values; optimal 1.0.
- Starting node P is winning (gives optimal value) for player to move.
- Limit is K errors...

1. below P, or

Fault Tolerance in Alternating Game Setting

- Given a chess engine E that may make up to K errors, can we use it to play perfectly?
- Perfect $\equiv$ plays any move that wins in a won position, or doesn't lose in a drawn position. Lost position—no restriction.
- Program can play against itself and return a value $v = E^*(P)$ from any position P.
- Can usefully abstract as "optimal" and "sub-optimal" values; optimal 1.0.
- Starting node P is winning (gives optimal value) for player to move.
- Limit is K errors...

1. below P, or

Smullyan-Style Liar Puzzle

- Proof that we can play perfectly given E makes at most 1 error below P.

Smullyan-Style Liar Puzzle

- Proof that we can play perfectly given E makes at most 1 error below P.
- Convert into induction on K...

Smullyan-Style Liar Puzzle

- Proof that we can play perfectly given E makes at most 1 error below P.
- Convert into induction on $K \ldots$
- $F(E, Q, j) = E'$ such that if E makes at most $j - 1$ errors below Q, then E' plays perfectly below Q.

Smullyan-Style Liar Puzzle

- Proof that we can play perfectly given E makes at most 1 error below P.
- Convert into induction on $K \ldots$
- $F(E, Q, j) = E'$ such that if E makes at most $j - 1$ errors below Q, then E' plays perfectly below Q.
- Can we modify the proof to work for criteria 2–4?