Theorem Proving and Model Checking
(or: how to have your cake and eat it too)

Joe Hurd
joe.hurd@comlab.ox.ac.uk

Cakes Talk

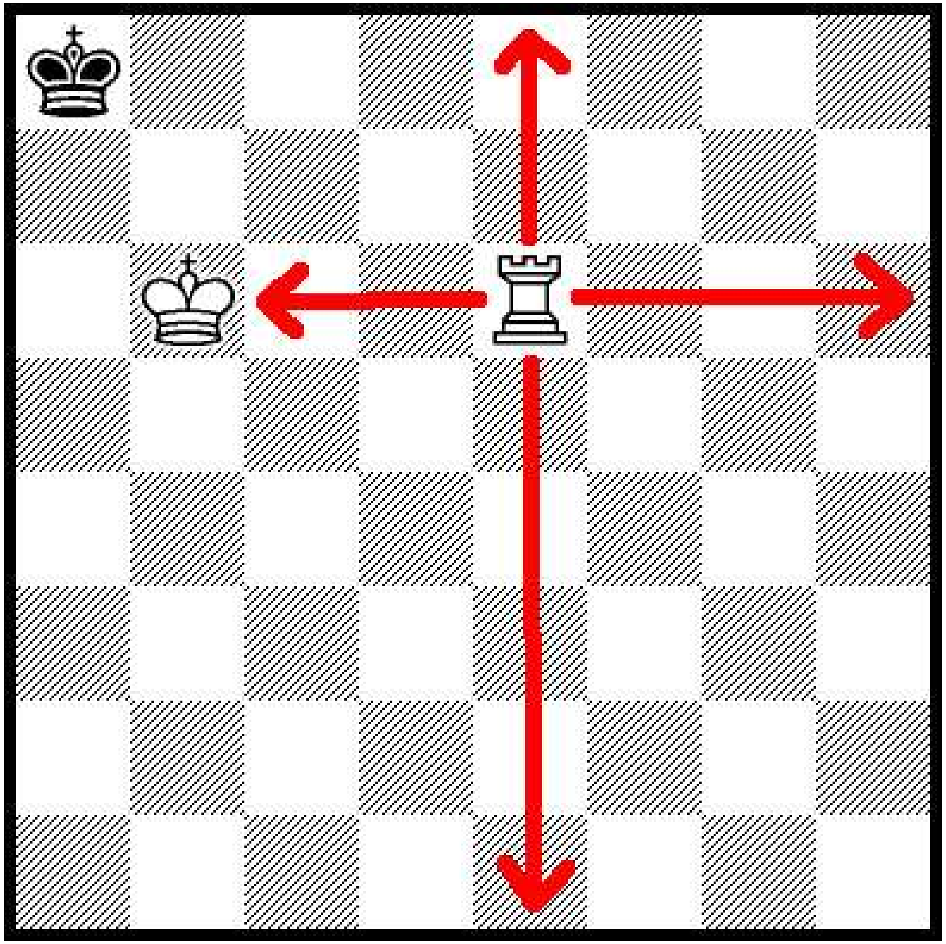
Computing Laboratory
Oxford University

Theorem Proving

- LCF-style theorem proving emphasizes high assurance.
  - Theorems can only be created by a logical kernel, which implements the inference rules of the logic.
- Higher order logic is expressive enough to naturally define many concepts of mathematics and formal language semantics:
  - probability via real analysis and measure theory;
  - the Property Specification Language for hardware.
- The main challenge is proof automation.

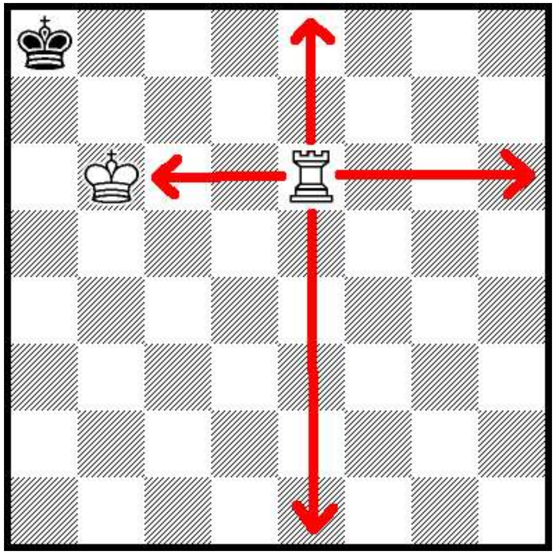
 $\rightsquigarrow$  $\rightsquigarrow$ 

Theorem Proving



Example: define the set of squares that a rook attacks.

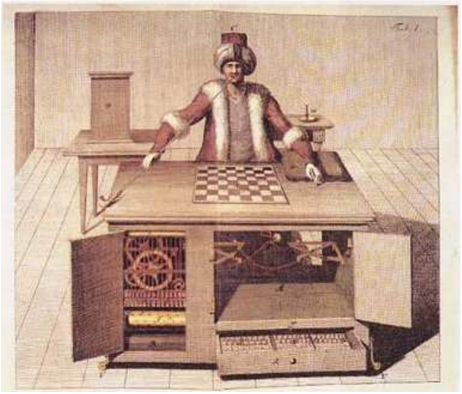
Theorem Proving



- square $\equiv \mathbb{N} \times \mathbb{N}$
  position $\equiv$ side $\times$ (square $\rightarrow$ (side $\times$ piece) option)
- rook_attacks $p\ a\ b \equiv$
  $a \neq b\ \wedge\ ($file $a =$ file $b\ \vee$ rank $a =$ rank $b)$
  $\wedge\ \forall c.$ square_between $a\ c\ b \Rightarrow$ empty $p\ c$
- The other rules of chess are similarly easy.

Model Checking

- Model checking emphasizes automation.
  - Various efficient algorithms for deciding temporal logic formulas on finite state models.
- High level input languages support the modelling and checking of complex computer systems:
  - IEEE Futurebus+ cache coherence protocol.
- The main challenge is to reduce problems to a form in which they can be efficiently model checked.



Combination Methods

- Approach 1: add theorem proving techniques to model checkers:
  - disjunctive partitioning of transition relations;
  - assume-guarantee reasoning;
  - data abstraction.
- This approach allows state of the art model checkers to tackle intractably large or even infinite state spaces.

Combination Methods

- Approach 2: implement model checking algorithms in theorem provers.
- Gordon created a set of inference rules relating higher order logic formulas and BDDs:

$$\frac{[a_1] \vdash t_1 = t_2 \qquad [a_2]\ t_1 \ \mapsto \ b}{[a_1 \cup a_2]\ t_2 \ \mapsto \ b}$$

- Amjad implemented a modal μ-calculus model checker called *HolCheck* as a derived inference rule in HOL4.
  - The resulting theorems depend only on the inference rules of HOL4 and the BuDDy BDD engine.
  - Used to verify several correctness properties of the AMBA bus architecture.

Verification Scripting Platform

- Higher order logic is a common semantics in which to embed many logics.
- HOL4 can be used a scripting platform to implement verification tools.
  - Pro: No error-prone translation between tools.
  - Con: Performance penalty for implementing as a HOL4 derived rule (about 30% for *HolCheck*).
- Example: using a formalization of PSL semantics to translate hardware properties to Verilog monitors.
- This talk: using a formalization of the rules of chess to construct a verified chess endgame database.

Chess Endgame Databases

- Can solve certain classes of chess endgame by enumerating all positions in a database.
  - Compute depth to mate by working backwards from the checkmate positions.
- Correctness is summed up by the following quotation:

  Both [Nalimov's endgame databases] and those of Wirth yield exactly the same number of mutual zugzwangs [...] for all 2-to-5 man endgames and no errors have yet been discovered.

- Ideally, we'd like to prove that the endgame database logically followed from the rules of chess.

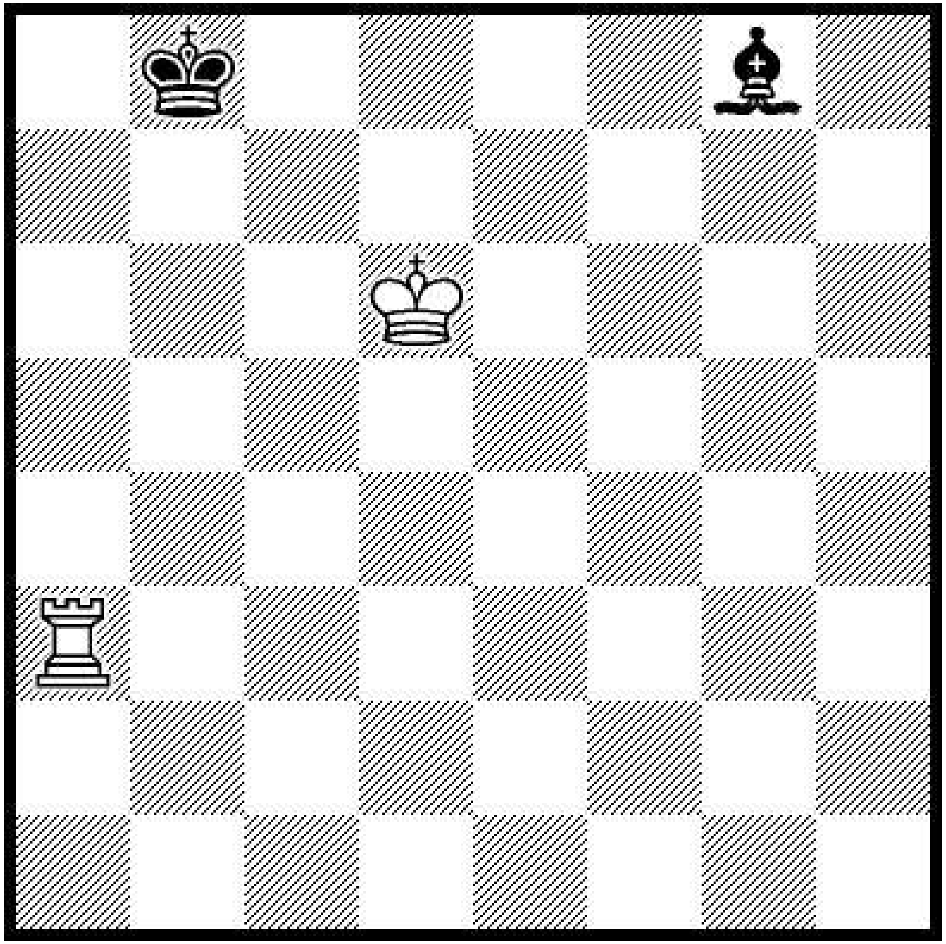
Verified Endgame Databases

We build our verified endgame database by working backwards from checkmates, but symbolically using BDDs.

$$
\begin{aligned}
[]\quad & \mathsf{abstract} \\
& \quad (\mathsf{decoder} \\
& \qquad (\mathsf{posn\_coder} \\
& \qquad\quad (\mathsf{Black}, [(\mathsf{White}, \mathsf{King}); (\mathsf{White}, \mathsf{Rook}); \\
& \qquad\qquad\qquad (\mathsf{Black}, \mathsf{King}); (\mathsf{Black}, \mathsf{Bishop})])) \\
& \qquad [b_0; b_1; b_2; b_3; b_4; b_5; b_6; b_7; b_8; b_9; b_{10}; b_{11}; \\
& \qquad\ b_{12}; b_{13}; b_{14}; b_{15}; b_{16}; b_{17}; b_{18}; b_{19}; b_{20}; b_{21}; b_{22}; b_{23}]) \\
& \in \mathsf{win2\_by}\ \mathsf{chess}\ 28 \\
& \longmapsto
\end{aligned}
$$

<29,907>

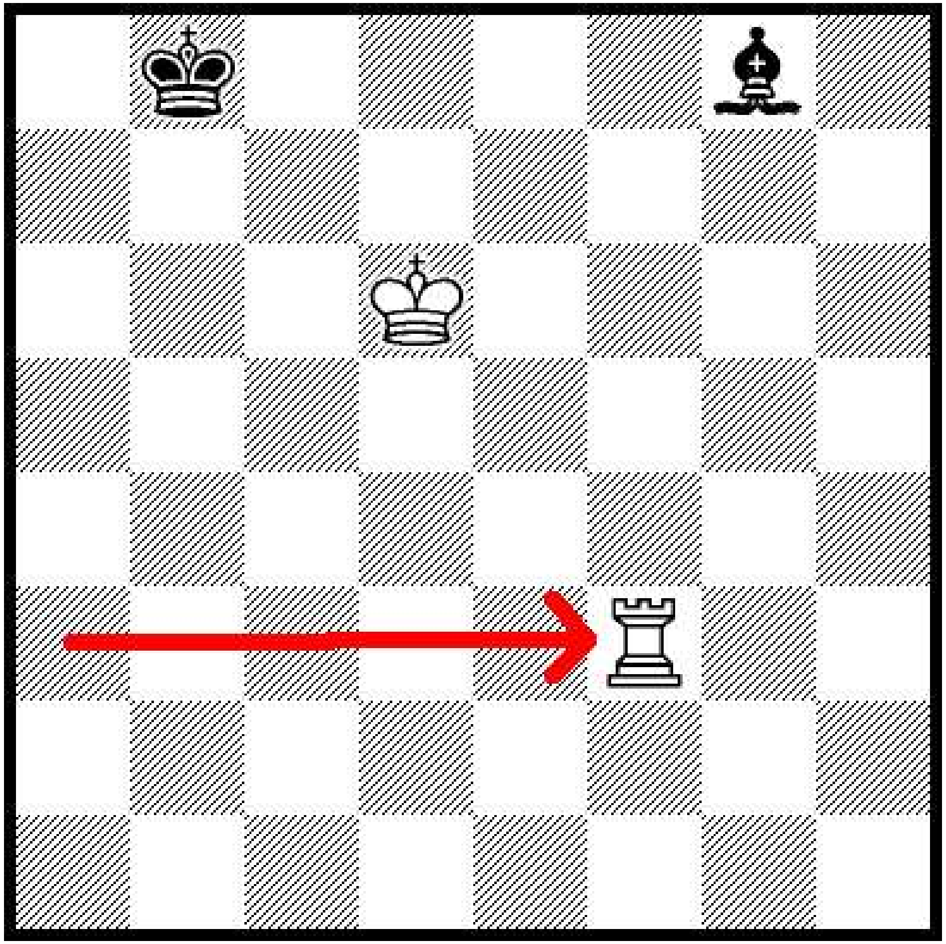
Verified Endgame Databases



One White move is checkmate in 29, all others draw.

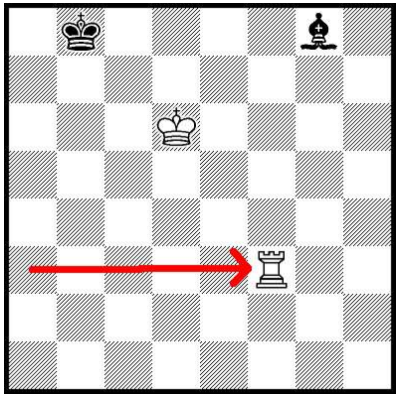
What is the winning move?

Verified Endgame Databases



Rf3!!

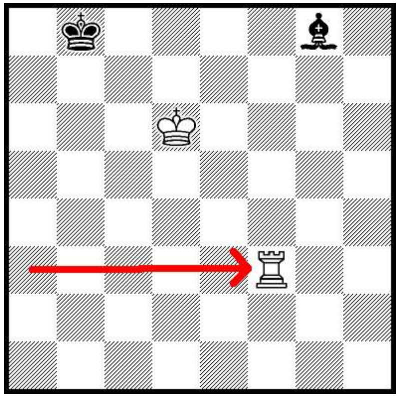
Verified Endgame Databases



The result of querying our verified endgame database on this position:

$$
\begin{aligned}
\vdash\ & (\mathsf{Black}, \\
& \lambda x. \\
& \quad \mathsf{if}\ x = (3,5)\ \mathsf{then\ SOME}\ (\mathsf{White}, \mathsf{King}) \\
& \quad \mathsf{else\ if}\ x = (5,2)\ \mathsf{then\ SOME}\ (\mathsf{White}, \mathsf{Rook}) \\
& \quad \mathsf{else\ if}\ x = (1,7)\ \mathsf{then\ SOME}\ (\mathsf{Black}, \mathsf{King}) \\
& \quad \mathsf{else\ if}\ x = (6,7)\ \mathsf{then\ SOME}\ (\mathsf{Black}, \mathsf{Bishop}) \\
& \quad \mathsf{else\ NONE})\ \in\ \mathsf{win2\_by\ chess}\ 28\ \wedge\ \cdots
\end{aligned}
$$

Verified Endgame Databases



In fact, checkmate in 29 is the longest possible win in the King and Rook versus King and Bishop endgame.

$$
\begin{aligned}
\vdash\ & \forall p. \\
& \quad \mathsf{all\_on\_board}\ p\ \wedge\ \mathsf{to\_move}\ p = \mathsf{White}\ \wedge \\
& \quad \mathsf{has\_pieces}\ p\ \mathsf{White}\ [\mathsf{King}; \mathsf{Rook}]\ \wedge \\
& \quad \mathsf{has\_pieces}\ p\ \mathsf{Black}\ [\mathsf{King}; \mathsf{Bishop}]\ \Rightarrow \\
& \quad p \in \mathsf{win1}\ \mathsf{chess} \iff p \in \mathsf{win1\_by}\ \mathsf{chess}\ 28
\end{aligned}
$$

Conclusion

- The first verified chess endgame database.
  - Query results logically follow from the rules of chess.
- Created by a combination of theorem proving and model checking.
  - Implemented as a HOL4 derived rule (with BDDs).
- Can solve all four piece pawnless endgames without any performance tuning.
  - Ken Thompson solved most five piece endgames, and the state of the art is now six piece endgames.
- Have put up some educational web pages showing the best lines of defence.
  - Checkmating a bare King with King, Bishop and Knight is something that beginners struggle to learn.