Proving Correctness of a KRK Chess Endgame Strategy by SAT-based Constraint Solving

Marko Maliković, University of Rijeka, Croatia
Predrag Janičić, University of Belgrade, Serbia

COST IC0901 Meeting — SVARM 2013
Madrid, Spain, October 17-18, 2013.

Introduction

- Within COST Action IC0901: a number of SAT-based systems and applications
- To be presented: application of SAT in chess, using the URSA system
- Marko Maliković, Predrag Janičić:
  Proving Correctness of a KRK Chess Endgame Strategy by SAT-based Constraint Solving.
  ICGA Journal (International Computer Games Association), 36(2) (2013).

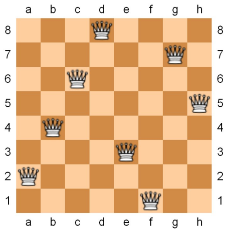
Overview

- Reduction to SAT and URSA system
- Chess endgame strategies and Bratko's strategy for KRK
- URSA specification of KRK and the strategy
- Correctness of the strategy
- Discussion and conclusions

Reduction to SAT

- Wide range of applications: planning, scheduling, operations research, combinatorial optimization, software and hardware verification
- SAT solvers — ,,Swiss army knife"
- ,,Efficient SAT solving is a key technology for 21st century computer science." (Clarke)
- Most often reduction to SAT is performed by ad-hoc tools

Example: Eigth Queens



- Different encodings can be used
- For instance: p_{ij} is true iff there is a queen on (i, j)
- Or: v_i is a row of the queen in i-th column (represented by three bits)
- Or: ...

Overview of URSA

- **U**niform **R**eduction to **SA**t
- Stand-alone, implemented in C++, open-source
- C-like specification language
- Unknowns are represented by bit-vectors
- Symbolic execution of specifications
- Constraints translated to SAT instances
- Models give solutions of the constraints

URSA: Example specification (Seed)

Linear congruential pseudorandom number generator:

```
nX = nSeed;
for(nI = 0; nI < 100; nI++)
   nX = 1664525 * nX + 1013904223;
assert(nX==123);
```

URSA: Properties

Theorem: If the variables v_1, v_2, …, v_n are (the only) unknowns in an URSA specification $S; assert(b);$,
then it leads to a solution
$(v_1, v_2, \ldots, v_n) = (c_1, c_2, \ldots, c_n)$,
iff
$\langle v_1 = c_1; v_2 = c_2; \ldots; v_n = c_n; S; assert(b), s_\emptyset \rangle \mapsto \langle skip, s \rangle$ where $s(b)$ is *true*.

Eight Queens Example

```
nDim=8;
bDomain = true;
bNoCapture = true;
for(ni=0; ni<nDim; ni++) {
   bDomain &&= (n[ni]<nDim);
   for(nj=0; nj<nDim; nj++)
      if(ni!=nj) {
         bNoCapture &&= (n[ni]!=n[nj]);
         bNoCapture &&= (ni+n[nj]!=nj+ n[ni]) && (ni+n[ni] != nj+n[nj]);
      }
}
assert(bDomain && bNoCapture);
```

Chess Endgame Strategies

- Different from midgame computer strategies
- Different from lookup tables approach
- Should be concise and intuitive and useful both for computers and humans
- Endgame strategies do not need to ensure optimal moves

(Simplified) Bratko's Strategy for KRK (for White)

Mate: If possible, mate black in two moves;

Squeeze: Otherwise, if possible, further constrain the area to which the black king is confined by the white rook

Approach: Otherwise, if possible, move the king closer to the black king

KeepRoom: Otherwise, if possible, maintain the present achievements in the sense of Squeeze and Approach

Divide: Otherwise, if possible, obtain a position in which the rook divides the two kings vertically or horizontally.

Modified Bratko's Strategy for KRK – No Search

- Instead of **Mate**:

  ImmediateMate: If there is such, play a mating move;

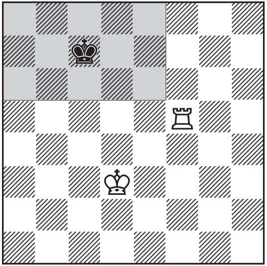
  ReadyToMate: If the above is not possible, then play a move that leads to the ,,ready to mate position"

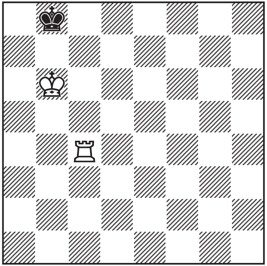
- Instead of **Divide**:

  RookHome: Otherwise, if possible, move the rook to be safe and horizontally or vertically adjacent to the white king

  RookSafe: Otherwise, if possible, move the rook to be safe and on some edge

Basic Ideas of the Strategy





Reducing ,,room" / ready for mate

Strategy Performance

The strategy is not optimal:

The number of positions with certain number of moves to win

Specification of Position

Bit-vectors of length 19 were used.

```
procedure Cartesian2Pos(nWKx, nWKy, nBKx, nBKy, nWRx, nWRy,
bWhiteOnTurn, nPos) {
  nPos = ite(bWhiteOnTurn,1,0);
  nPos = (nPos << 3) | (nWRy & 7);
  nPos = (nPos << 3) | (nWRx & 7);
  nPos = (nPos << 3) | (nBKy & 7);
  nPos = (nPos << 3) | (nBKx & 7);
  nPos = (nPos << 3) | (nWKy & 7);
  nPos = (nPos << 3) | (nWKx & 7);
}

procedure Pos2Cartesian(nPos, nWKx, nWKy, nBKx, nBKy, nWRx, nWRy,
bWhiteOnTurn) {

...
```

Specification of „Black is Attacked“

```
procedure BlackKingAttacked(nPos, bBlackKingAttacked) {
  call Pos2Cartesian(nPos, nWKx, nWKy, nBKx, nBKy, nWRx, nWRy,
bWhiteOnTurn);
  call Between(nWRx, nWRy, nWKx, nWKy, nBKx, nBKy, bBetween);
  bBlackKingAttacked = (nWRx==nBKx ^^ nWRy==nBKy) && !bBetween;
}
```

Specification of ImmediateMateCond

```
procedure ImmediateMateCond(nPos1, nPos2, bImmediateMateCond) {
  call LegalMoveWhiteRook(nPos1, nPos2, bLegalMoveWhiteRook);
  call Mate(nPos2, bMate);
  bImmediateMateCond = bLegalMoveWhiteRook && bMate;
}
```

Example Lemma

Lemma: Starting from any legal KRK position, after a step by white (by strategy) and a legal step by black, the obtained position is again a legal KRK position.

Indeed, no model for:

```
call LegalKRKPosition(nPos1,bLegalKRKPosition1);
call StrategyStep(nPos1,nPos2,b1,nStep1);
call LegalMoveBlack(nPos2,nPos3,b2);
call LegalKRKPosition(nPos3,bLegalKRKPosition3);
bKRKLegalityNotPreserved = (bLegalKRKPosition1 && b1 && b2 &&
!bLegalKRKPosition3);

assert(bKRKLegalityNotPreserved);
```

Relation $\rightarrow$

Definition [relation $\rightarrow$]: $p_1 \rightarrow p_2$ holds iff:

- it is white's turn in p_1 and p_2 is reached from p_1 following a strategy step or;
- it is black's turn in p_1 and p_2 is reached from p_1 by a legal ply.

Partial Correctness

Theorem [Partial Correctness]: If a KRK game ends, then the last move was made by white and black is mated.
Follows from:

- **Lemma**: If a KRK game ends, then the last ply was made by white.
- **Lemma**: If, after a ply by of white, black cannot move, then black is mated (not stalemated).

Termination

Theorem [Termination]: The relation $\rightarrow$ is well-founded.
Follows from

- A suitable measure (decreases in 6-plies) and
- Several lemmas ...

Discussion: Size and Hardness

- The largest lemma: 463770 vars/1641425 clauses
- The hardest lemma: 1.6h (using the Clasp solver)

Discussion: Clarity

- High-level statements vs. exhaustive analysis
- URSA can be used for both
- In many mathematical contexts, high-level arguments are preferable
- Given URSA specifications are high-level and readable

Discussion: Target Theory

- The lemmas are translated into SAT instances
- ... but could be also translated to BVA or LA instances
- Our systems URBIVA and URSA Major can do that

Discussion: Confidence

- The (high-level) lemmas can be also checked by a C program.
- However, systems like URSA are more reliable (since the solving process is delegated)

Discussion: Confidence (2)

- Lemmas are "proved" individually, cannot be glued together into URSA statements
- Solution within a proof assistant would give higher confidence
- Without support for SMT provers, it is often difficult to construct hard combinatorial proofs within proof assistants
- The current support for SMT solvers within Coq cannot handle the generated formulae (subject of current work)

Conclusions

- An approach for proving properties of chess procedures presented
- Can be used in other domains
- We will also use translation to LA and Coq for similar approaches.