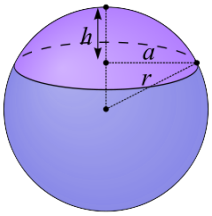
Home * Engines * **CAPS**



Spherical Cap [5]

CAPS, (Chess as Problem Solving, CAPS-II)
Hans Berliner's experimental chess program which was subject of his 1974 Ph.D. thesis *Chess as Problem Solving: The Development of a Tactics Analyser* at Carnegie Mellon University [1], further eloborated at the Advances in Computer Chess 1 conference [2], and, along with B*, in a Panel on Computer Game Playing [3]. CAPS is a selective, knowledge driven depth-first alpha-beta searcher in the domain of chess tactics. A position is represented as vector of 1040 words of information, including the list of generated pseudo-legal moves, kept on a stack of up to 20 ply.

CAPS-II was tested on many middle-game chess tactics problems from standard textbooks on chess. It played a few complete games of chess. In both these modes it ran with a maximum depth of 10 ply. CAPS-II did not do too well in the games, since it had little positional knowledge, and more importantly the tactics mechanisms were still not completed, causing it to make occasional blunders which would wipe out whatever good it had achieved earlier. However, it did quite well on the tactics problems such as Reinfeld's Win at Chess [4].

Table of Contents

Refutation Description

While visiting a node, CAPS determines attack information, and whether pieces are completely or partial en prise or overprotected, are pinned, or may be source of a discovered attack, and further collects a so called **Refutation Description** of expandable sets (bitboards and piece-sets) of moving and captured pieces, their source- and target squares, and appropriate paths of sliding pieces involved, associated with the full qualified moves (piece, from, to, captured piece if any) along the current search line. The description of the six sets of the Refutation Description is given from the Advances in Computer Chess 1 paper [6]:

- **RPCS** - is a bit-vector which has bits representing names of pieces.
  The name bit of the piece that moved to produce this node is set in this vector.
- **RSQS** - is a bit-vector with bits representing squares on the board.
  The bit corresponding to the destination square of the move that produced this node is set in RSQS.
- **RPATH** - is a bit-vector with bits representing squares on the board.
  The bit for any square across which a sliding piece moved in making the made move is set in RPATH. If the move was a non-

capture pawn move, then all squares over which it passed including the destination also have bits set for them.

- **RTGTS** - is a bit-vector with bits representing names of targets. A comparison is made of BEST for this node with BEST one ply previously. Any squares which are now named as containing material targets, but were not mentioned in the previous BEST, have bits set for the name of the piece on this square to indicate that this threat was created by the last move.
- **TGTSQS** - is a bit-vector with bits representing squares on the board.
  For any RTGTS detected as above, bits are set in TGTSQS for the corresponding squares.
- **TPATH** - is a bit-vector with bits representing squares on the board.
  For any TGTSQS detected as above, if a piece that has an ATTACKING function on this square is a sliding piece, then all the intervening squares have bits set for them in TPATH.

Causality Facility

While classical depth-first searchers may perform the killer heuristic, or may analyze the backed-up principal variation for refutations, i.e. to move or defend any captured piece, or capture or pin any capturer, or block the path of any moving piece, CAPS' refutation description allows for much more complete understanding of a set of consequences, not only restricted to the PV. The **Causality Facility** interprets the refutation descriptions and tries to determine tactical causes and goals to control a set of move generators for possible goal transitions, and allows pruning of large sub-trees. The facility may further execute a null move to determine whether a suspected move was responsible for a bad position or not.

Method of Analogies

Once a move was determined as in fact bad, it is possible to posit a Lemma along with its environment of the refutation description — this being knowledge that a certain move will be bad as long as the described environment does not change. With this knowledge, any proposed move may be looked up in the Lemma file and if it has been previously cataloged, the program may determine if the current position contains any essential changes from the Lemma environment which might make the move succeed. It is important to note that should it be decided to try the move, and should it again fail, that it would now be possible to generalize the Lemma to include the union of the two environments, thus making it stronger. In a somewhat similar way, it is possible to generalize about the movements of a single piece, if more than one Lemma exists with respect to its moves. Lemmas are also posited about winning moves. Whenever a move is suggested at some future point in the search, the lemma file is first examined to determine if a lemma exists about this move and if the partial board description matches. If so, the result of the move is assumed

known and the search can be foregone. A similar system has been described by Georgy Adelson-Velsky, Vladimir Arlazarov and Mikhail Donskoy [7] [8].

Quotes

Hans Berliner on CAPS-II in his Oral History, March 2005 [9]:

And here we are in 1970 and the Northwestern people have just made a big splash and everybody realized that they were very, very good and that they were not the kind to rest on their laurels, they kept finding things to improve. Now I was at Carnegie-Mellon where I had been admitted at the age of 40, not because of any academic credentials but because I knew how to do chess and I'd already written a computer program and they had hopes that I could push this state of the art doing computer chess, which I tried to do. But my program was not in the style of these modern programs, it was a program based on conceptual things and it was a very - a lot of effort was devoted to structure; in other words every position had a lot of structure and there were programs that delineated the structure ...

There was one thing in this dissertation that I'm proud of and that was something that I called 'the causality facility' and - this is a dead end though I have to say that ahead of time, it was a dead end but it was a nice one - and the idea was that when, lets say, a human being makes a certain move and then a rook comes down to the back rank and says 'checkmate' the human being says 'oh I gotta do something about that, I can't make a move over here and he's gonna give me checkmate.' And I was once having a conversation with Minsky and he said you know 'how do humans do this and why don't machines do this?'

So I started thinking about that and when I did my dissertation I had descriptions - as I said I had a lot of structure - and I dragged descriptions back as one backed up from some terminal node and you backed the value not only do you back the value up, you're backing up a description and the description said which pieces moved where and - and some other information. So in other words at some point you would arrive at the point where you say 'oh, this last move must have been a mistake because I got checkmated' and then you look at this description and see what the opponent did and say 'oh probably I will lose again the same way unless I do something about that description' which meant you either had to capture the piece that's doing the moving or you had to block it or guard the square on which it landed or something like that. And that's what my program was able to do, and it did so me very nice clever things where something was being

```
threatened and it figured out the only way to block it without trying
everything, by just reasoning about what the characteristics of a move
would have to be in order to prevent this, so it was doing - it was
good Ph.D. work but it didn't fit into the main scheme . Well I think
as a knowledge exercise it was good.
```

See also

- Acronym
- Paradise

Publications

- Hans Berliner (**1973**). *Some Necessary Conditions for a Master Chess Program.* 3. IJCAI 1973
- Hans Berliner (**1974**). *Chess as Problem Solving: The Development of a Tactics Analyser*. Ph.D. thesis, Carnegie Mellon University
- Hans Berliner (**1977**). *A Representation and Some Mechanisms for a Problem-Solving Chess Program.* Advances in Computer Chess 1
- Hans Berliner, Richard Greenblatt, Jacques Pitrat, Arthur Samuel, David Slate (**1977**). *Panel on Computer Game Playing*. 975-982 5. IJCAI 1977, pdf
- Editor (**1979**). *Chess Ratings from Problem Solving*. Personal Computing, Vol. 3, No. 12, pp. 77 » Chess Problems, Compositions and Studies

Forum Posts

- Re: Method of Analogies?? by Steven J. Edwards, CCC, May 29, 1998
- Re: Symbolic: Status Report 2007.04.25 by Steven Edwards, CCC, April 26, 2007 » Symbolic

External Links

Caps

- Caps (disambiguation) from Wikipedia
- CAPS entreprise - many-core programming
- Caps lock from Wikipedia
- Calcyphosin (CAPS gene) from Wikipedia
- Computer Animation Production System - Wikipedia
- Java Caps from Wikipedia » Java
- Problem solving from Wikipedia

Cap

- Cap (disambiguation) from Wikipedia
- Cap from Wikipedia

References

1. ^ Hans Berliner (**1974**). *Chess as Problem Solving: The Development of a Tactics Analyser*. Ph.D. thesis, Carnegie Mellon University
2. ^ Hans Berliner (**1977**). *A Representation and Some Mechanisms for a Problem-Solving Chess Program.* Advances in Computer Chess 1
3. ^ Hans Berliner, Richard Greenblatt, Jacques Pitrat, Arthur Samuel, David Slate (**1977**). *Panel on Computer Game Playing*. 975-982 5. IJCAI 1977, pdf
4. ^ Hans Berliner (**1977**). *A Representation and Some Mechanisms for a Problem-Solving Chess Program.* Advances in Computer Chess 1
5. ^ An example of a spherical cap by Pbroks13, October 2008, Spherical cap from Wikipedia
6. ^ Hans Berliner (**1977**). *A Representation and Some Mechanisms for a Problem-Solving Chess Program.* Advances in Computer Chess 1
7. ^ Georgy Adelson-Velsky, Vladimir Arlazarov, Mikhail Donskoy (**1975**). *Some Methods of Controlling the Tree Search in Chess Programs*. Artificial Intelligence, Vol. 6, No. 4, pp. 361-371. ISSN 0004-3702. Reprinted (**1988**) in Computer Chess Compendium pp. 129-135
8. ^ Method of Analogies?? by Bruce Cleaver, CCC, May 29, 1998
9. ^ Oral History of Hans Berliner, Interviewed by: Gardner Hendrie, Recorded: March 7, 2005, The Computer History Museum, pdf, pp. 25.

What links here?

Page	Date Edited
Advances in Computer Chess 1	Oct 22, 2015
CAPS	Dec 23, 2017
Carnegie Mellon University	Feb 12, 2018
Chess Problems, Compositions and Studies	Feb 19, 2018
Engines	Mar 10, 2018
Hans Berliner	Jun 10, 2017
Herbert Simon	Feb 11, 2016
Paradise	Nov 19, 2014
Personal Computing	Dec 27, 2017
Playing Strength	Mar 31, 2018
Tactics	Jan 12, 2018
Test-Positions	Feb 25, 2018
William Chase	Jan 21, 2016

Up one Level