A THESIS IS MORE THAN THREE UNRELATED IDEAS

New Architectures in Computer Chess
Fritz M.H. Reul
Reviewed by Dap Hartmann

Even though computers have surpassed the playing strength of the best human chess players, it remains an interesting research area both from a computer science point of view as well as from the chess perspective. With ever increasing computer playing strength, human chess players will continuously be surprised by the intricacies of chess positions which they thought they understood. While it appears that computers have little more to learn from humans, human chess players may still learn many things from computers. However, there is a limit to what human chess players can learn and comprehend. That already became clear with the appearance of the first endgame databases which contradicted human judgement. For example, it was thought that KBBKN was a drawn endgame (provided that Black could build a fortress) until Ken Thompson's database proved that it was acutally a win for White. So, there lies an enormous challenge ahead in programming computers to teach humans how to improve their chess skills. Early experiments by John Roycroft showed that it is quite difficult for human chess players to improve their endgame skills by merely consulting an omniscient database. Whether it will be at all possible to translate the implicit knowledge of how to win complicated endgames into a format that humans can understand, remains to be seen. It is certainly an interesting research area.

With this future for computer chess in mind I looked at the project that Fritz Reul undertook for his Ph.D. thesis. His main research objective was to 'develop new computer-chess architectures that adequately deal with the complex chess knowledge'. I interpreted that as the intention to develop the tools which will be necessary to embark upon the aforementioned endeavour. Therefore, I was surprised by the first of Reul's three research questions: "To what extent can we develop non-bitboard computer-chess architectures, which are competitive in speed, simplicity, and ease of implementation?" That implies that there is a need for an alternative to bitboards and raises the question: what is wrong with bitboards? Strangely enough, that question is neither raised nor answered. Reul goes back to basics and revisits the internal representation of a chess board. He discusses Minimal Board Borders and argues that the upper and lower board borders must be at least two squares wide to avoid a Knight from jumping off the expanded board or jumping around the board (off the board on one side and back onto the board on the other side). That surprised me because in my own very first chess program (DAPPET, written in FORTRAN in 1980), I used a linear array of 120 elements to represent the expanded chess board. The 8×8 actual chess board was embedded in the expanded board which measured 10×12 squares. There were two extra rows below and above the real board, and one extra file on each side. Every piece except the Knight would eventually run off the edge of the chess board into the expanded board. Knight moves (jumps) were covered too. The eight offsets of a knight move ('knight increments' in Reul's terminology: -21, -19, -12, -8, +8, +12, +19, +21) when added to any square index of the chess board [21..28, 31..38, …, 91..98] always indexed a square of the expanded board. No jumps outside the expanded board and no moves around the chess board. Yet, Reul claims that the internal chess board must have a minimum dimension of 10×15 squares, 25% larger than what DAPPET used 30 years ago.

In Chapter 2, Reul compares one of his New Architectures with a Rotated Bitboard architecture. In four middle game positions and four endgame positions, the New Architecture was consistently faster by 17% on average for doing and undoing all legal moves 10 million times (performing all administrative tasks such as updating piece lists and board indexes). Maybe it was implicit or maybe I overlooked it somehow, but what I missed is the time required to generate the legal moves in the first place. Moreover, I had preferred to read earlier that the challenge of the Chapter was to pave the way for a comparison with *bitboards* (see Chapter 3) instead of with *rotated bitboards*. This was mentioned at the end of Chapter 2. One of the interesting advantages of using bitboards is that it allows you to generate only the moves for pieces that were affected by the previous move, while all other moves remain valid. This may be less efficient in terms of time and memory use than generating all moves anew (and only when you need them) but it is definitively more elegant.

The New Architectures can certainly be beneficial to games with boards that have more than 64 squares, such as Shogi (9x9), at least until 128-bit CPUs are commonplace. But the title of this thesis suggests application to the game of chess and therefore I found it a bit surprising that the first research question deals with finding alternatives to the highly efficient (rotated) bitboard architecture. The idea to represent the chess board by 64 bits is almost as old as the very idea of using a computer to play chess, and I would have liked to see an overview of the historical development of bitboards in computer chess. However, Reul takes Bob Hyatt's rotated bitboards as the state-of-the-art starting point and the architecture to challenge.

Apparently, even for Reul abandoning the rotated bitboard was a surprising research outcome. Therefore, in Chapter 3 he addresses as second research question: "To what extent is it possible to use hash functions and magic multiplications in order to examine bitboards in computer chess?" After showing that his New Architecture outperforms a rotated bitboard architecture, the author continues to focus on bitboards. The results of this Chapter have a highly-empirical nature (Reul points to LOOP's second place in the 26th Open Dutch Computer-Chess Championship (2006) and the third place in the 15th WCCC (2007)) and are accompanied by many recommendations for future research.

Reul's third research question reads: "How can we develop an αβ-approach in order to implement pruning conditions in the domain of static exchange evaluation?" It appears unrelated to the first two research questions. In my opinion, only one of the three research questions bears any resemblance to the main problem statement "How can we develop new computer-chess architectures in such a way that computer-chess engines combine the requirements on knowledge expressiveness with a maximum of efficiency?" And even though he does present a new efficient computer-chess architecture (second requirement), I could not discover any capability in knowledge expressiveness (first requirement). It appears that Reul's thesis is a report of three successful ideas that he developed while creating his chess program LOOP. The problem was to find a common denominator, a suitable umbrella to sell it as a coherent research project. Reul has not succeeded in that, and his thesis therefore lacks coherence and focus.

The last proposition in Reul's list of propositions that traditionally accompanies a Dutch Ph.D. thesis is rather revealing: "Writing a thesis is more exhausting than developing a state-of-the-art computer-chess engine." While undoubtedly true, only a programmer-turned-researcher would state that so explicitly. It seems to me that Fritz Reul is a talented computer-chess programmer who has set himself the task of writing a PhD thesis on three of his ideas and on their implementations. The result is a piece of work that distinguishes the scientist from the clever programmer.

First movement: Carmen Leza (General Manager of CEIN), Jaap van den Herik, Jose María Roig (Minister of Navarra Government)