A status report on Losing Chess

Mark Watkins
University of Sydney,
School of Mathematics and Statistics
watkins@maths.usyd.edu.au

It's unfortunate that so much knowledge seems to have gone missing. – Gian-Carlo Pascutto

1. INTRODUCTION

The purpose of this writing is to report on the current status of Losing Chess. Since the naming of this class of games has variant schools of terminology, we state for definiteness that captures are compulsory (but the choice of the player on turn when there are multiple captures), that the King has no special characteristics, and that castling is not legal. Promotion to a King is allowed. This is also sometimes called Antichess.

In particular, we give information about responses to 1. e3. Some of this dates back more than a decade, and is due to others. However, other parts described herein have a few novel features. Unless otherwise stated, we use the International Rules, so that a stalemate is a win for the player on move.[1]

1.1 Historical sources

There exist a number of piecemeal Internet sites that have various information about Losing Chess. However, many of the pages of interest have not been touched in 5-10 years. Most of them use FICS rules, for which stalemate is a win for the player with fewer pieces (and drawn when the piece counts are equal). See also the ICGA page [H].

1.1.1 Pages of Fabrice Liardet

Fabrice Liardet is one of best (human) Losing Chess players in the world. His French language site [Li] has a wealth of information about the game. It seems that these pages were last modified in 2005. The most directly relevant pages for our discussion are the opening pages on 1. e3.

1.1.2 Pages of Cătălin Frâncu

Cătălin Frâncu is the author of the Nilatac program [F1]. The last changes seem to be no later than 2006 (though see §3.5). He also provides a wonderful browseable opening book [F2]. He is now working on a next-generation solver, named Colibri.

1.1.3 Pages of Vladica Andrejić

Vladica Andrejić has a page that dates from 2004 [A]. While his Encyclopaedia of Suicide Chess Openings is not very complete (at least in the online version), it does list some historical information unavailable elsewhere.

[1] If one is trying to prove that White wins via 1. e3, the *vinciperdi* ruleset, where stalemate is a draw, would dominate both International and FICS Rules. However, this game tends to take on a much different flavour, as Black can try to draw via a strategy of blocking enough pawns, for this greatly increases the difficulty of White to lose all the pieces. One can note that almost always a stalemated side has nothing but pawns remaining.

For instance, it notes that ASCP (the program of Ben Nye) refuted the Andryushkov Defence (1. e3 c6 2. Bb5 cxb5 3. b4!) on Feb 3, 2003.

1.1.4 Pages of Carl Lajeunesse

As far as I can determine, `suicidechess.ca` [La] is the only currently active site for information about Losing Chess. It has an immense opening book (around a billion nodes, with 15% of these expanded), each corresponding to a significant pn-search. The efficacy of this book is unclear, as it only proved 1. e3 c6 on Dec. 6,‘2009. His website interface provided much of the inspiration for the interface described in §2.3 below.

1.2 Our results

We announce here that 1. e3 Nc6 and 1. e3 b5 are both won for White. To the best of our knowledge, these are new results. However, we cannot say that our techniques show forth much innovation. These results could presumably have been computed many years ago. Much of our work was simply greater effort and patience.

Our overall node counts are still much smaller than those for the solution of checkers [Sch], and our work is easier on various accounts. For instance, the difficulty involved in proving a win is typically less than in proving a draw,[2] and in particular we did not have to worry various side issues, particularly with transpositions (though these were still a headache at times).

We also announce that 1. e3 Nh6 is a win, though our proof follows that of Frâncu (see §3.5). We finally announce that 1. e3 g5 is a win, with part of our proof again building on Nilatac's book.

1.3 Acknowledgements

In addition to the above webpages, the author would like to thank Gian-Carlo Pascutto and Lenny Taelman for useful comments.

2. SEARCH METHODOLOGY

Here we briefly describe our search methodology. One could describe the method in terms of PN^2-search [BUH], but I personally think a different explication is more useful. We consider proof-number searches to be akin to an evaluation function. That is, for a given position P we search a certain amount of nodes in pn-search, for us, either 10^7 nodes or 10 seconds, the former more typically occurring first, but the latter sometimes being the cut-off in endgame positions. The other termination criteria in the pn-search included: side to play has no move, only 4 units left (tablebases), repetition of position, and draw by lone bishop with opponent having one of opposite colour.

This pn-search then returns two values, the proof and disproof numbers for P. An evaluation for P is determined from them, for instance as the ratio of two (this is an idea of Frâncu), and then this is then stored with P in a higher-level tree. The local results of the pn-search below P are largely discarded, though we also included (recursively starting at P) a child node whose subtree (by a crude count) was at least 70% as that of the parent.

We then allow the upper-level tree to grow, extending leaf nodes via pn-search in some best-first manner, such as minimaxing the ratios/evaluations up to the root and expanding a critical leaf.

However, to introduce some randomness into the selection process, we in essence took an idea from how opening books randomise their choices. We walked down the upper-level tree from the root node choosing a child randomly according to a weighting from its minimaxed ratio (typically as the square of it, so that a child of ratio 25.0 would be 4 times more/less likely to be followed as one with ratio 50.0). This idea also makes it easier to parallelise the scheme – just run N independent instances of the pn-search on various leaf nodes.

[2] As an aside, we might note that draws need not be much more difficult: for instance, if each side is forced to repeat moves, under pain of a fairly quick loss. But the impression that draws are generally harder to prove than wins still does seem to have some merit.

2.1 Transpositions

For transpositions, in both the pn-search and the upper-level tree we chose to identify nodes with the same position only when the reversible move count was zero. This had the advantage of dispelling any loops, though of course it is not very optimal. One improvement could be to also identify nodes which are known wins/losses, but this already gets a bit tricky in repetitions, and I never found the energy to implement it. Presumably one could be more highbrow in a couple of standard situations, namely forced *en passant* (such as both g3 and g4 having only hxg3 as a response) and multiple promotions when the promoting unit must then be captured, but I never found that the effort (and care) involved was likely to be rewarding.

In our node counts, we counted each unique position only once. The alternative method would be to count each path-expansion, no matter how many times the underlying position occurs in the tree, and no matter whether we identified such nodes in the tree. One reason that this latter method might be preferred could be that the arcs of the graph are labelled by moves, and these are typically stored (in the data structure) on the target node.

To give an idea of how this affects our counts, the final proof tree for 1. e3 b5 had 108066622 nodes, and 5349608 transposition pointers. Of the nodes themselves, 99761505 were internal nodes, and 8305117 were terminal nodes, with 1962171 of the latter being in 4-unit tablebases.[3]

The large percentage of internal nodes is typical of Losing Chess, as often a win will contain a long sequence of forced Black moves, which when alternated with solitary White moves, creates long chains of single-child nodes.

2.2 Performance of automated methods

The above formed the basis of our automated search process. Typically we ran it on a 6-processor machine (so that six upper-level nodes were being expanded at any given time), and produced about 1 million upper-level nodes in an overnight run. Due to various factors (such as disk caching in the operating system), allowing the upper-level tree to get much larger than about 4-5 million nodes (about 200MB in our implementation) led to poor performance. Thus we developed some tools designed to help cut and splice these upper-level trees.

One disadvantage of the ratio-expansion is that one can sometimes wander into a "well" where almost all White moves have a great advantage, but none easily lead to wins. This is typical when White has a large material advantage, and Black a lone king, but White needs to push a pawn (or two) to promotion before the final wipeout. Another quirk is that there is a rather notable tempo-advantage in ratio-expansion. Often a upper-level node with unexpanded children will have its ratio go up/down by a factor 5-10 (or more) upon expanding them. I was not able to come up with any easy solution to this, and its interaction with the 70% child-inclusion from pn-nodes (which would tend to alternate who was on move, particularly upon a forced capture) was another difficult aspect.

After some initial experimentation, we found it quite advantageous to declare a draw to be a win for Black. This had the advantage of clearing up a lot of repetition draws from the upper-level trees. Adding knowledge of the lone opposite bishop draw was also useful, and of course tablebases (described more below) were quite powerful. However, we would still occasionally run into the "well" problem of above, and indeed the final solving of a upper-level tree would often take much longer than might be expected.

2.2.1 Some possible improvements

One idea that we never implemented (at either the upper-level or in the pn-search) was a killer heuristic at sibling nodes; for instance, one could change the move priorities via modifying the evaluation from the ratios.

Another idea (compare enhanced transposition cutoffs [ETC]): upon creation of a upper-level node, look at its possible children, and see if any (via transposition) are already known to give a result that proves the node. This should be easy to implement, but again I never found the motivation to do so. A similar task could be to avoid dominated lines. A final consideration is that the predictive value of ratio seems related to game phase, that is, a ratio of 100.0 with many pieces left on the board will often be solvable rather quickly, but the same ratio in a

[3] With respect to Footnote 1, of the 6342946 terminal positions not from tablebases, in 2521246 of them White had no pieces, and in the remaining 2812700 White had pawns remaining but was stalemated. Of these latter, only 4051 would not be wins under FICS rules. I did not try to obtain similar statistics for the tablebase positions.

situation where Black has only a King and pawns is rather likely to just be a slow endgame win.

2.3 Human input

There were two major ways that the above search procedure was augmented by human input. The first was in the choice of which upper-level tree would be the next to be explored and/or cut or spliced. In some of the more difficult lines, we would have upper-level trees (each with more than a million nodes, each node from a pn-search of size 10^7) corresponding to sub-sub-sub-sub-variations, which tended to make the project rather onerous from the standpoint of data management.

The second enhancement was via a Java-based interface that allowed the user (namely myself) to choose what upper-level node to expand next. This displayed the proof/disproof numbers and ratio, and allowed one to queue up to about 64 positions to be searched. As noted above, this was inspired by the `suicidechess.ca` website of Carl Lajeunesse.

As a rough estimate, the solving of 1. e3 b5 took about 2-3 cpu-years, about 2-3 work-months using the Java-based interface (some of which was rather mindless clicking, but most of it seemed motivated), and about the same amount of human time in code development, including learning enough about Losing Chess and previous programs so as to have an idea of how to proceed. The Java code itself was adapted from the "ComradesGUI", written by the developers of IPPOLIT (see below).

2.4 Tablebases

The efficacy of having at least some tablebases can be seen from the position with (say) a White King on d8 and a Black pawn on d5. This is a draw (Black will King the pawn), though the ratio from a pn-search of 10^7 nodes is around 300, as White has much more mobility (hence many more options) than Black, at least at first.

We thus decided to develop a program to generate tablebases for Losing Chess. This, of course, is not novel, though I could not find anything particular to International Rules that had been done. At first, we decided to adapt the RobboBases of IPPOLIT developer "Roberto Pescatore" (this seems to be a pseudonym). However, in the end we ended up being unable to use almost all the clever ideas it contained: for instance, the index-differencing was found to be too dependent on the king-structure of normal chess, so we chose to re-compute the index from scratch. Similarly, with the king-slicing unavailable, the SMP machinery was then seen as too unwieldy, especially as we had no plans to build 6-unit TBs. The concept of a BlockedPawn counting as one unit was also dumped, even though it should be even more valuable in Losing Chess (where pawns on adjacent files can also be counted in such a way, with a bit more work).

In the end, our code built the 4-unit TBs in around 2 hours, and the 5-unit TBs in a couple of weeks. A verification unit detected a few errors (with *en passant*, unsurprisingly), but these were then fixed.[4] Following the RobboBases, we decided to use distance-to-conversion as the metric. In the pn-search, only the 4-unit TBs were accessed, and these were read from a flattened array of 2 bits per entry (WDL or broken). This takes about 800MB of memory, and allows fairly fast access.

In the human input stage, we would sometimes access the 5-unit TBs (which were only built rather latterly) to determine if a line was worth pursuing. If so, then we typically could copy over the relevant moves manually in about 5-10 minutes, with the upper-level tree then having resolved the position.

The 5-unit TBs could presumably be accessed in the pn-search, at least near the root, via a compression scheme such as that described in [TB]. For normal chess, this reduces the size of the 5-unit TBs to about 450MB (both the Shredderbases and the RobboTripleBases are about this size, so too more recent work of Ronald de Man), at the cost of some additional computational overhead in capture resolution. It is unclear to me what the comparative size for Losing Chess would be; firstly, there are more endgames (as the king is no longer royal), and secondly the compression efficacy from capture resolution would be likely differ (due to captures being mandatory). As one goal of our research was to provide final proof trees which needed only the 4-unit TBs, we did not pursue this avenue too deeply.

[4] As above, I do not know of any other source for Losing Chess TBs for International Rules. However, since we are proving wins rather than draws, an alternative method of verification is simply to expand all relevant tablebase positions until a terminal position is reached.

2.4.1 Some 5-unit tablebase facts

The longest 5-unit tablebase loss[5] under DTC occurs in KPkrp, at 78 moves. Others in the 2-vs-3 genre with a loss longer than 50 are KNkkr and KNkkn (both 54) and KRkkn and KNkrn (both 55). In the 1-vs-4 genre, there is Kkbnp at 74, Kqnnp at 61, Kqbnp at 60, and nine others with a win taking more than 50 moves. In the Appendix, we give two maximal 5-unit positions, and a mainline for each.

These can be compared to the longest 4-unit conversions, where the Kkbn configuration already has a loss in 71 (wKd1 bKa3 bBf6 bNa5), with Kkkn at 50, and 7 others above 40 moves. None of these have pawns (except for Rknp), while almost all the long 1-vs-4 losses contain a pawn for the winning side (Kbnnn at 43 moves is the longest that does not). It does not seem that adding a fifth piece increases the complexity of conversion that much, at least when compared to normal chess.

The Appendix additionally has some examples of full-point zugzwangs, again comparing to the 4-unit case.

Ben Nye (Angrim) computed such 5-unit tablebases for FICS rules about a decade ago, and Ronald de Man (syzygy) also appears to have done so.

2.5 Post-processing the finished trees

The desired result of the above process would be a (pruned) upper-level tree which was completely solved. One then still needs, however, to expand this into a full proof tree. At this stage, we re-traversed the upper-level tree (after identifying all identical positions, irrespective of the reversible move count), and then re-ran the pn-search solver on each leaf node. In a perfect world, this would suffice to re-solve the node – however, due to various gremlins (for instance, the pn-search takes as input a specific path to a position, and so its internal expansion procedure might slightly differ upon transposition), this was not always the case.

The re-expansion part of our suite of programs was for a long time underdeveloped, but eventually we enhanced it to allow automatic fixing of failed expansions. A typical upper-level tree would have 1 million nodes from the search driven by ratio-expansion and human input; this would reduce to around 25 thousand nodes upon pruning, which would then expand to around to about 5 million nodes in a full proof tree (the re-expansion process itself would not identify node transpostions from different pn-solutions, but a utility to do this was instrumented). These typical numbers could vary by a factor of 2-3 or more, depending on the type of position (such as whether or not long endgames resulted). These full proof trees themselves could then be spliced together, if desired.

Our upper-level trees were stored in a ratther bulky data format, using about 40 bytes per node. Each node had proof/disproof numbers, fields for parent, child, and sibling, and a somewhat hackish implementation of transpositions that used two fields. These were combined with a field for ratio (which was not saved to disk, but was computed on loading), a field to estimate the subtree size (not always used, but occasionally useful during a pruning stage to decide which White move to retain when multiple wins were known) and a half-field (16 bits) for the move played to reach this node. Another half-field was reserved for various extraordinary usage (such as marking a node as won/lost without any backing from search).

This allowed easy updating of proof/disproof numbers, sorting of children by ratio, and management of transpositions.[6] The final proof trees use 6 bytes per node, in a more compressed format. For instance, if node N has children, the first child must be node N+1 (with then node N+1 pointing to any sibling, and so on). A similar method was used with next-siblings when a position transposes, and the child and transposition flags are themselves 1-bit fields above a 30-bit node-number indicator (the other 2 bytes are to record the move that corresponds to the incoming arc). This does not allow easy manipulation of trees, and tasks such as backtracking from a given node to the root require some extra work. The main advantage of this format is that it is more condensed than the fuller format.

[5] This is a remnant of how the RobboBase code works – it saves whether a position is won, drawn, or lost-in-X. It also does not distinguish whether White or Black makes the conversion.

[6] We used the same data structure in the pn-search, which meant that our standard pn-search of 10^7 nodes took about 400MB. Multiplying this by a factor of 6 for our trivial parallelisation, and adding the 800MB for 4-unit TBs, our typical overnight job would fit comfortably in 4GB, though as noted above, various operating systems could make life difficult with disk caching, and in fact we generally had a 8GB machine in any event.

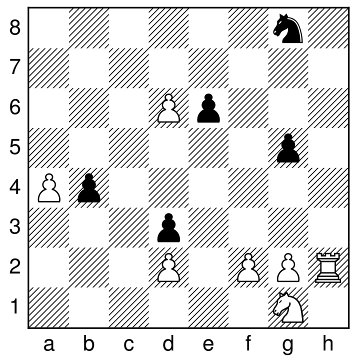
2.6 Final tree verification

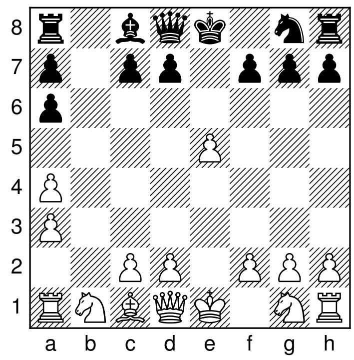
An important component of our suite of programmes is the verifier. This checks a number of tree properties, such as whether all Black moves are considered, whether hash-identified nodes are the same, that terminal nodes are wins (White has no moves, or the final position has 4 units and is a White win), and more. This found a number of problems at various stages of our work. The major disadvantage from an independence standpoint is that it uses the same move generation and tablebases as the main programme.[7] Frâncu has been able to transfer our proof of 1. e3 Nc6 into FICS rules with Nilatac, giving another partial verification of our work.

The LosingGUI described in §2.3 was also adapted into a WinningGUI, that allows one to walk through a proof tree, also giving counts on the size of subtrees.

3. SOLVED RESPONSES TO 1. E3

3.1 The well-known





	U	final
f6	87	273036
a5	152	261691
a6	302	234355
f5	26	89635
h5	24	69190
e5	7	43271
h6	28	43130
Nf6	1	23388
g6	1	4489
Na6	1	3271
d6	1	33
d5	1	33

Table 1: Simplification in 1. e3 c6, mainline of 1. e3 Nc6, and node counts for 12 simple responses to 1. e3.

Black has 20 responses to 1. e3, with 12 of these are fairly easy to refute. Two of them, namely d5 and d6, are particularly trivial. All of these are folklore, having been known for some time. In Table 1 (right), we give the size of the trees we obtained; see §2.1 for more about accounting, though it should be said the accounting of an upper level nodes (here denoted U) has not always remained constant throughout our work. We make no claims of the minimality of the final node counts of our proof trees.

3.2 1. e3 c6 (Andryushkov Defence)

As noted above, this seems to have been first solved by Ben Nye's program ASCP in February 2003, and a solution also exists in Nilatac's opening book. We largely copied over Nilatac's tree manually, but also found a simplification in the position given in Table 1 (left). This arises after 1. e3 c6 2. Bb5 cxb5 3. b4 b6 4. Ke2 a5 5. bxa5 bxa5 6. c4 bxc4 7. Kd3 cxd3 8. a4 Na6 9. e4 Qc7 10. e5 Qxc1 11. Qxc1 Rb8 12. Qxc8 Rxb1 13. Rxb1 Nb4 14. Qxe8 g6 15. Qxf7 e6 16. Qxd7 Bd6 17. Qxh7 Rxh7 18. exd6 Rxh2 19. Rxh2 g5 20. Rxb4 axb4. After 21. g4, Black has either b3 or e5 to put up any real resistance. We found that 21. g4 e5 22. a5! wins much more easily than Nilatac's 22. Rh7. Meanwhile, the try 21. g4 b3 22. Rh7 e5 23. Rb7 e4 24. Rxb3 Nf6 25. Rxd3 exd3 26. a5 Nxg4 27. Nh3 Nxf2 28. Nxf2 g4 29. Nxd3 g3 succumbs rather easily after 30. a6 (rather than 30. Nb2).

One aspect is that Frâncu solved this in conjunction with Nye's 5-piece tablebases, which could yield quite bulky trees when expanded. Klaas Steenhuis was able to verify (using Lajeunesse's website) that our simplifications largely transferred to FICS rules.

[7] Much of our move generation code followed that of IvanHoe (again from the IPPOLIT developers), adapted suitably for Losing Chess.

3.3 1. e3 Nc6 (Balkan Defence, according to Andrejić)

To the best of my knowledge, this was not previously proven to be a loss. However, it is not really *that* much more difficult than 1. e3 c6. The automated searching process yielded an abnormally large proof/disproof ratio after about a cpu-week of running time, and then about 10-15 hours of manual work finished the job.

The main line 1. e3 Nc6 2. Ba6 bxa6 3. a4 Nd4 4. exd4 has about 81% of the nodes, then 4. . .e5 has about 3 times as large a subtree as 4. . .Nh6. Black's first non-majority choice is move 6 (after 5. dxe5 Ba3 6. bxa3, see Table 1 center): here Qh4 (26%), Nh6 (21%), a5 (13%), Nf6 (12%), Kf8 (11%) and h5 (8%) all have large subtrees.

3.4 1. e3 b5 (Classical Defence)

Again the previous status of this opening is unclear to me. When I told Pascutto that I was close to solving it, he seemed to remember seeing a lecture about its resolution (or something related) some years ago.

Black here has three main tries, two of which then themselves split into three more main lines. We can note in passing that the "Suicide Defence", namely 1. e3 b5 2. Bxb5 Bb7, has long been known to be losing for Black.

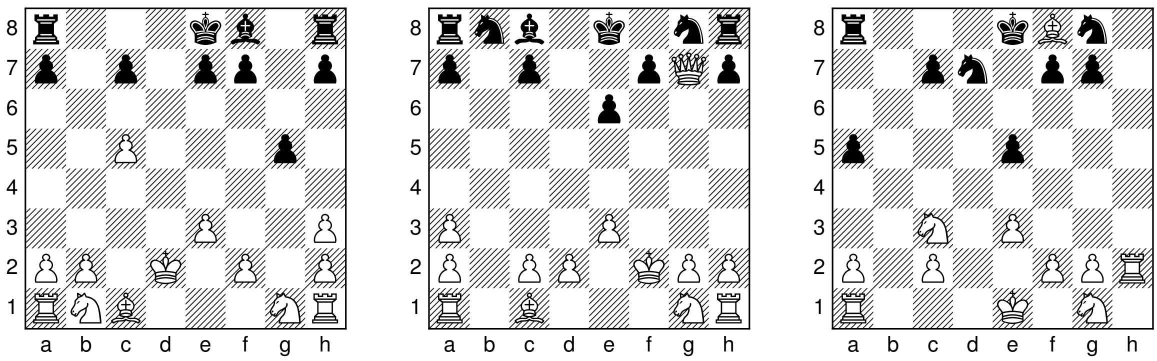


Figure 1: Mainlines in 1. e3 b5

3.4.1 1. e3 b5 2. Bxb5 Nh6

This defence is already about as complicated as 1. e3 Nc6 with our proof subtree having 16057049 nodes. About 61% of it concerns Black capturing on d7 with the Knight. The mainline is 1. e3 b5 2. Bxb5 Nh6 3. Bxd7 Nxd7 4. c4 Ng4 5. Qxg4 g5 6. Qxd7 Qxd7 7. c5 Qxd2 8. Kxd2 Bh3 9. gxh3 (Figure 1 left), whereupon Black has a number of choices. Both Bh6 and g4 have subtrees of around 2.5 million nodes, while Kd8 is around 1.5 million, and Rg8, Kd7, and Rc8 are all around 500 thousand or more.

3.4.2 1. e3 b5 2. Bxb5 e6

This proof subtree has around 28.4 million nodes. After 1. e3 b5 2. Bxb5 e6 3. Bxd7 Bxd7 4. Na3 Bxa3 5. bxa3, Black can complicate matters with Qh4, c6, or Bc8.[8] Both of the latter are about 20-25% of the subtree, while Qh4 leads to 6. Qg4 Qxf2 7. Kxf2 Bc8 8. Qxg7 (Figure 1 center) when Black has either a5 or Ne7 that have 4 million nodes or more, with a6 and Na6 also over 1.5 million.

3.4.3 1. e3 b5 2. Bxb5 Ba6

This is Black's most lasting defence. After 1. e3 b5 2. Bxb5 Ba6 3. Bxd7 Nxd7 4. d3 Bxd3 5. Qxd3, each of Rb8, h6, and particularly Qb8 take substantial effort to defeat. The mainline of the latter, still having over 32 million nodes, is 6. Qxh7 Rxh7 7. Nc3 Qxb2 8. Bxb2 Rxh2 9. Rxh2 a5 10. Ba3 e5 11. Bxf8 (Figure 1 right), when either recapture leads to a subtree of 18-19 million nodes. In either case, White plays Rh6, Black captures with the

[8] An unpublished guide to openings by Andrejić says Bc8 was the mainline before ASCP (of Ben Nye) proved it to be a loss, via 6. a4 Qxd2 7. Bxd2 (opposed to our 7. Kxd2). No dates are given.

pawn, and then White plays f4 followed by a pawn exchange. This reduces it an endgame where both sides have a King, a Rook, two Knights, and four pawns. Perhaps a bit surprisingly, White is sufficiently better co-ordinated so as to win.

3.5 1. e3 Nh6 (Hippopotamus Defence)

Of the 7 unsolved lines from when this project began, Nilatac's book gave this one the highest proof/disproof ratio (that is, most likely to be a win). In fact, when we announced our results on 1. e3 b5 privately (on August 31, 2012), Cătălin Frâncu responded that he had recently shown (while testing a new laptop) that this line was indeed won for White under FICS rules (where stalemate is a win for the player with fewer pieces), taking about two cpu-months of computing time.

His upper-level tree had 668 thousand nodes, which reduces to 167 thousand upon transposition-detection. We instrumented a utility to transfer his tree to our set-up. Upon simply attempting to solve all the upper-level nodes (including internal ones), this taking about 12 cpu-hours, we were left with only 15 unsolved nodes,[9] and less than 10 minutes of manual work gave us a solved tree.[10] This was then expanded into a full proof tree as in §2.5, with the final node count being 19118683.

After 1. e3 Nh6 2. Ba6 bxa6 3. Qh5, Black has either g6 or c5, and c6 also lasts over 2.3 million nodes. In the first line, 3. Qh5 g6 4. Qxg6 fxg6 lasts 3 times as long as hxg6, the principal follow-up being 5. Ne2 Kf7 6. Na3 a5 7. g4 Nxg4 8. Rg1 Nxf2 9. Rxg6 Kxg6 10. Kxf2 a4 11. Nf4 Kh5 12. Nxh5 Bg7 13. Nxg7 Qe8 14. Nxe8 Rxe8 15. b3 axb3 16. cxb3, and Black's queenside is too undeveloped to survive for long. The other mainline is 3. Qh5 c5 4. Qxh6 gxh6 5. b4 cxb4 6. Ba3 bxa3 7 .Nxa3 Qc7 8. Nh3 Qxc2 9. Nxc2 Bg7 10. Ke2 Bxa1 11. Rxa1 a5 12. a4 Na6 13. f3 h5 14. Ra3 h4 15. Kd1 f5 16. g3 hxg3 17. hxg3 Kf7 18. Ke1 Rf8 19. Kf2 Rh8 20. g4 fxg4 21. fxg4 Rd8 22. Kg3 (Figure 2 right), and again Black's material advantage does not offset the lack of piece co-ordination (as can be seen by the Rook shuffles on the last few moves).

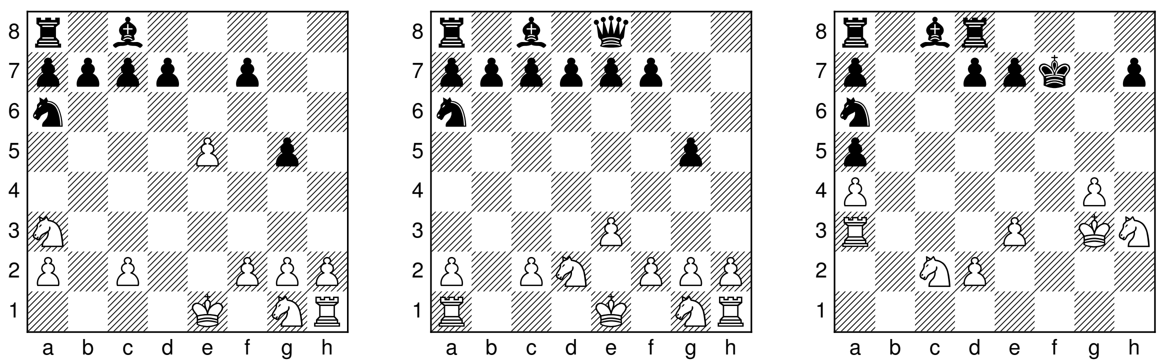


Figure 2: Mainlines in 1. e3 g5 2. Ba6 Nxa6, and in 1. e3 Nh6

3.6 1. e3 g5 (Wild Boar Attack)

With the above aid from Frâncu fortifying us that there might still be some relatively easy lines left to prove, we turned to 1. e3 g5, which had generally not seen much analysis. We first looked at 2. Bd3, but upon noting that 2. Ba6 bxa6 had been solved by Nilatac, we switched to this. Black's alternate try of 2. Ba6 Nxa6 took under a week to solve (about a cpu-month), with the final overall proof tree weighing in at 55594000 nodes.

Almost 90% of the node-count is in the Nxa6 line, and there are two main variations after 3. Qh5 Bg7 4. Qxh7 Bxb2 5. Qxh8, when both Bxa1 and Bxc1 have subtrees over 22 million nodes. In the first line, White plays 6. Qxg8, and then the Black's toughest defense is Kf8 (54%), with Bc3 also over 5 million nodes, and various other moves over a million. The mainline is then 6. Qxg8 Kf8 7. Qxf8 Qxf8 8. Bb2 Bxb2 9. d4 Bxd4 10. exd4 e5 11. dxe5 Qa3 12. Nxa3 (Figure 2, left), when b5 takes over 5 million nodes to defeat, while c5 is over 3.5 million, and Nc5 is also nearly 1.5 million.

[9] Our upper-level tree had only 25000 nodes, as our pn-solver was able to solve many nodes higher in the tree than Nilatac was, thus making the subtrees redundant.

[10] We tested our machinery by first resolving 1. e3 c6 via similar importation of Nilatac's tree; in that case, our final proof tree had about 100000 nodes less than our first proof.

If Black instead captures with 5...Bxc1, then the mainline is 6. Qxg8 Bxd2 7. Qxe8 Qxe8 8. Nxd2 (Figure 2, center), with b6, Nb8, and Qf8 all over 6.5 million nodes, and c5 also above 2.7 million, not to mention that f6 is over a million, and c6 and d6 each over 600 thousand. White meets b6 or Nb8 with f4 (and these often transpose), while Ne2 defeats Qf8, with again Black's two main moves being b6 and Nb8, the former met by f4 and the latter by Nb3 (in which case Black again prefers b6, with White playing Kd2, forgoing f4 in this sequence).

The transfer of Nilatac's proof for 2. Ba6 bxa6 saw no problems. The mainline here is 3. Qh5 Bh6 4. Qxf7 Kxf7 5. e4 Qe8 6. e5 Bf8 7. Ne2 Nf6 8. exf6 exf6 9. a4 Qxe2 10. Kxe2, where both Ba3 and a5 have around 500 thousand nodes or more.

3.7 Table of node counts of various lines

Line	nodes
All 12 easy lines	1045130
1. e3 c6	2356831
1. e3 Nc6	12121049
1. e3 Nh6	19118683
1. e3 g5	55594000
1. e3 b5	108066622
1. e3 g5 2. Ba6 bxa6	6213822
1. e3 g5 2. Ba6 Nxa6 3. Qh5 Bg7 4. Qxh7 Bxb2 5. Qxh8 Bxa1	23477913
1. e3 g5 2. Ba6 Nxa6 3. Qh5 Bg7 4. Qxh7 Bxb2 5. Qxh8 Bxc1	22257581
1. e3 b5 2. Bxb5 Ba6	59683294
1. e3 b5 2. Bxb5 e6	28435433
1. e3 b5 2. Bxb5 Nh6	16057049
1. e3 b5 2. Bxb5 e6 3. Bxd7 Bxd7 4. Na3 Bxa3 5. bxa3 Qh4	14823407
1. e3 b5 2. Bxb5 e6 3. Bxd7 Bxd7 4. Na3 Bxa3 5. bxa3 c6	7338823
1. e3 b5 2. Bxb5 e6 3. Bxd7 Bxd7 4. Na3 Bxa3 5. bxa3 Bc8	5847461
1. e3 b5 2. Bxb5 Ba6 3. Bxd7 Nxd7 4. d3 Bxd3 5. Qxd3 Qb8	41495895
1. e3 b5 2. Bxb5 Ba6 3. Bxd7 Nxd7 4. d3 Bxd3 5. Qxd3 h6	10268374
1. e3 b5 2. Bxb5 Ba6 3. Bxd7 Nxd7 4. d3 Bxd3 5. Qxd3 Rb8	6040970

4. THE THREE REMAINING RESPONSES

We have made some effort to solve the 3 remaining responses (e6/b6/c5). In each case, it seems that White can enter an endgame a piece up, but if wins are possible, they seem to come quite tediously.

5. ELECTRONIC AVAILABILITY

We have set up a webpage for downloading our programmes, including tools for search, verification, and tree manipulation, and the GUIs for controlling search and exploring final proof trees (which are also available).
The URL is: `http://magma.maths.usyd.edu.au/~watkins/LOSING_CHESS`

6. WEB RESOURCES

[A] V. Andrejić, Suicide chess, http://poincare.matf.bg.ac.rs/~andrew/suicide/index.html

[F1] C. Frâncu, Nilatac program, http://catalin.francu.com/nilatac

[F2] C. Frâncu, Opening book, http://catalin.francu.com/nilatac/book.php

[H] G. Haworth, Losing Chess, http://ilk.uvt.nl/icga/games/losingchess

[La] C. Lajeunesse, http://suicidechess.ca

[Li] F. Liardet, Losing chess, http://www.pion.ch/Losing

7. BIBLIOGRAPHY

[BUH] D. Brueker, J. Uiterwijk, J. van den Herik, *The PN^2-search algorithm.* In *Advances in Computer Games 9*, edited by J. van den Herik and B. Monien, Universiteit Maastricht (2001), 115–132. ISBN 90-6216-5761 / 90-6216-5664.

[ETC] A. Plaat, J. Schaeffer, W. Pijls, A. de Bruin, *Exploiting Graph Properties of Game Trees.* 13th National Conference on Artificial Intelligence (AAAI-96), Vol. 1, pp. 234-239, ISBN 978-0-262-51091-2.

See also *Nearly Optimal Minimax Tree Search?*, Technical Report 94-19. Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada.

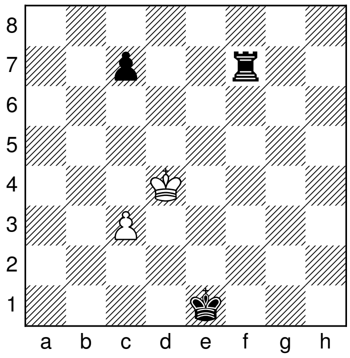
[Sch] J. Schaeffer, *Game Over: Black to Play and Draw in Checkers.* ICGA Journal **30** (2007), no. 4, 187–197.

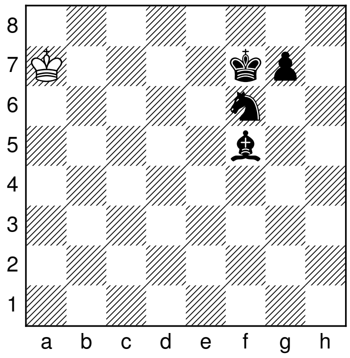
[TB] J. Schaeffer, Y. Bjornsson, N. Burch, R. Lake, P. Lu, S. Sutphen, *Building the Checkers 10-piece Endgame Databases*. In *Advances in Computer Games* **10** (2003), edited by J. van den Herik, H. Iida, E. A. Heinz, Advances in Information and Communication Technology Vol. 135, Kluwer, 193–210. ISBN 1402077092.

8. APPENDICES

APPENDIX A: MAXIMAL LINES IN 5-UNIT TBS (DTC METRIC)

Figure 3: White to play and lose in 78 (left), and in 74 (right)





1. Ke5! Rh7!! 2. c4! c5!! 3. Ke4! Rh1!! 4. Kf4! Kd1!! 5. Ke4! Kc1! 6. Kf3! Rd1 7. Kg2! Rd8 8. Kf1 Ra8 9. Ke2 Kb2! 10. Ke3 Ka3! 11. Ke4 Ka4! 12. Ke5! Rh8 13. Ke6 Rh1 14. Kd7 Ra1 15. Ke6! Ka3! 16. Ke5 Kb2! 17. Ke4 Kc1! 18. Kf3! Kc2! 19. Kg2! Kd1! 20. Kg1! Ra3! 21. Kh2! Ra8! 22. Kg2! Rd8! 23. Kg3 Kd2 24. Kg2 Rd6 25. Kg3! Rb6! 26. Kg2! Rb2! 27. Kg3! Rb1! 28. Kg4! Re1 29. Kg3! Rd1! 30. Kg4! Ke1! 31. Kf4! Kf1! 32. Kg4! Re1! 33. Kf5! Ke2! 34. Ke5! Rb1! 35. Kf5! Kf2! 36. Kg5! Re1 37. Kf6 Ke3! 38. Ke6 Rb1! 39. Kf6 Kf3! 40. Kg6 Kg3! 41. Kg7! Rg1 42. Kg6! Rg2! 43. Kf6! Rh2! 44. Kg7! Kh4! 45. Kh7! Re2! 46. Kg7! Kg4! 47. Kf8! Rh2! 48. Kf7! Rh1! 49. Ke7 Kh5! 50. Ke6 Rh2 51. Ke5 Kh6! 52. Ke6! Kh7! 53. Ke5! Rg2! 54. Ke6! Rg8! 55. Ke5! Rg7! 56. Ke4! Kh6! 57. Ke5! Rh7! 58. Ke4! Kh5! 59. Ke5 Rb7! 60. Ke4! Rb6! 61. Ke3! Kg6! 62. Kf3! Kf6! 63. Ke3 Ke6! 64. Kf3! Rd6! 65. Kf2! Kf5! 66. Ke1! Rg6! 67. Ke2! Rg8 68. Kd2 Kg4! 69. Kd3 Rb8 70. Kd2! Rb6 71. Ke1! Kf4! 72. Kd1! Ke4 73. Kc1! Rf6 74. Kb2! Rf4! 75. Ka1! Rf3 76. Kb1! Ke5 77. Ka1! Re3 78. Kb1 Kd5! 0-1

1. Kb7! Bh7!! 2. Kb6! Bb1!! 3. Kb5 Kg6!! 4. Kb4 Bf5!! 5. Kb5! Kg5! 6. Kb6! Bg6!! 7. Kb5! Kf4! 8. Kb4! Bf5!! 9. Kb5! Bg4! 10. Kb6! Ke5!! 11. Kb7! Bh5!! 12. Kb6! Kf5! 13. Kb5 Bg4!! 14. Kb4! Ke5 15. Kb5! Bh3! 16. Ka5! Bf5! 17. Kb4 Ng4 18. Kb3! Bc8!! 19. Kc2! Nf6! 20. Kb3! Bh3! 21. Kb2! Ng4! 22. Kc2! Kd5! 23. Kb3 Kd6! 24. Kc3 Nh6!! 25. Kd3 Nf7! 26. Ke2! Bd7! 27. Kd3! Ke6 28. Kc3! Bc8!! 29. Kd3! Ke7! 30. Ke3! Bd7! 31. Kd3! Kf8! 32. Ke3 Kg8! 33. Kd3 Bc8! 34. Ke3! Kh7! 35. Kf2! Kg6! 36. Kg2! Kf5!! 37. Kh3 Ke6!! 38. Kg3! Nd6! 39. Kf2 Bd7! 40. Kg2! Be8! 41. Kg3! Kf6! 42. Kf2 Ke5! 43. Ke2! Bd7!! 44. Kf2! Nf7! 45. Kg2! Be8! 46. Kh2! Nh8! 47. Kh3! Bf7! 48. Kh2! Ng6! 49. Kg2! Be8! 50. Kf2! Bd7! 51. Ke2! Bc8! 52. Kd2! Bb7! 53. Kc2! Kf4! 54. Kc3 Bg2 55. Kb4! Ke5! 56. Kb5! Bh3!! 57. Kb4! Bg4! 58. Kb3! Nf4! 59. Kb2! Bc8! 60. Kc1! Nh3! 61. Kd2 Bb7! 62. Kc2! Kf4! 63. Kc3 g5! 64. Kc4! Bc8!! 65. Kc5 Bf5! 66. Kc4! Bg4! 67. Kc5! Bd1! 68. Kd6 Kf3! 69. Ke6 Ke3 70. Kd6 g4! 71. Kc5 Kf3! 72. Kd6 g3! 73. Ke6 g2! 74. Kd6 g1=R (win in 7)

APPENDIX B: ZUGZWANGS

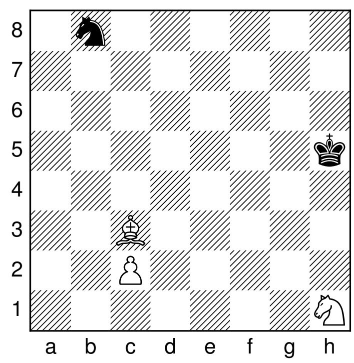
We can also list some full-point zugzwangs of interest. Already in the 4-unit TBs we have wKc5,bKa2,bKa1,bNb1 (Kkkn 45/2) as one where White takes 45 moves to lose (Black loses in 2, the idea here being that only a Kb2 move avoids immediate loss, but then White can play Kd4, and Black has c3 doubly attacked). With NNnn there is the symmetrical wNh8,wNa4,bNa1,bNh5 (NNnn 36/36) that loses in 36 for whomever is on move, and others such as wNg1,wNh2,bNa8,bNb7 (NNnn 31/23); indeed, often in Losing Chess one eliminates Knights and pawns from zugzwang accounting, as they tend to create too many. This leaves Rkkk with wRd3,bKb6,bKb1,bKg1 (Rkkk 20/7) as the longest at 20 moves, and if one excludes this gropuing, then there is nothing more than 7 moves, here wRa5,bKf3,bKc2,bBd1 (Rkkb 7/2) and wKd6,bQb1,bRf2,bBg1 (Kqrb 7/2). The symmetrical wKc2,wBb1,bKf7,bBg8 (KBkb 5/5), losing in 5, is also perhaps worth mentioning.

There are about 136000 total full-point zugzwangs, of which around 1000 have no knight or pawn.

APPENDIX B.1: 3-vs-2

The situation is similar in the 4-unit genre. When allowing knights and pawns, there is a loss in 37 from wRc4,wNa4,bKf6,bNh1,bPd7 (RNknp 37/3), and wBc3,wNh1,wPc2,bKh5,bNb8 (BNPkn 19/15, Figure 4) is the most notable example where both sides have a significant number of moves to make.

Figure 4: White loses in 19, Black in 15

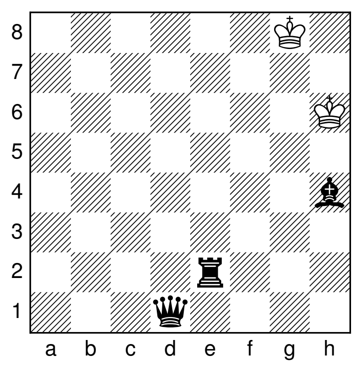


1. Ba1 Na6!! 2. Nf2! Kg5 3. Bb2 Nc7! 4. Ba3! Kf5 5. Bc5! Ke6! 6. Be3 Ne8 7. Nd3! Ke7 8. Nb4! Kf6! 9. Bg1 Kg5! 10. Ba7! Ng7! 11. Bg1 Nh5! 12. Bb6! Kg4 13. Ba5! Ng3 14. c3 Nf5 15. c4! Kg5 16. Nd3! Kf6 17. Nf2! Ne7 18. c5! Nc8! 19. c6 Nb6! 0-1

1... Kg4! 2. Bb4!! Kf5! 3. Nf2!! Ke6! 4. Bf8! Kf7! 5. Ba3! Kf6! 6. Bc5! Ke6! 7. Nd1 Kf5! 8. Nb2 Ke6! 9. Ba3 Ke5! 10. Bb4 Ke6! 11. Bc5! Kf5 12. c4! Kf6! 13. Bb4 Kf7 14. c5! Kf8! 15. Be1 Nc6 16. Ba5 1-0

Upon excluding knights and pawns from the mix of pieces, we have losses in 8 for wKc1,wKb2,wBa1,bRe7,bBf8 (KKBrb 8/1), wKc2,wKb3,wBb1,bKb5,bRe4 (KKBkr 8/1), and wKd8,wKc2,wBb1,bQh4,bRf6 (KKBqr 8/1). There is also wKc3,wBb2,bKg7,bRe5,bBh8 (KBkrb 6/4) and finally wKg8,wKh6,bQd1,bRe2,bBh4 (KKqrb 6/4, Figure 5) where both sides have a few moves to make before losing.

Figure 5: White loses in 6, Black in 4



1. Kgh7! Bg3!! 2. K6g6! Bd6! 3. Kh5! Qf1!! 4. K5h6! Qf2 5. Kh8! Qa7 6. Kh5 Qg7 0-1

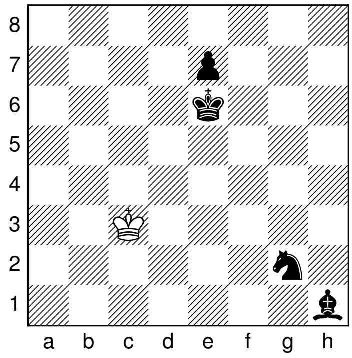
1... Be1 2. Kh5! Bf2 3. Kg7! Bg1 4. Kf8 Ba7 5. Ke7 1-0

Overall, there are approximately 540000 full-point zugzwangs here (an exact count is exacerbated due to symmetry quirks which I have yet to sort out), of which about 5300 have no Knight or pawn.

APPENDIX B.2: 4-vs-1

The longest-lost zugzwang is with wKc2,bQh4,bNf5,bNg3,bPc6 (Kqnnp 50/3), where White to move takes 50 moves to convert. There are two other positions over 40 moves: wKa3,bKb6,bBa6,bNb8,bPc6 (Kkbnp 45/2), and wKb3,bQd8,bNa8,bNf5,bPc6 (Kqnnp 44/4). A loss with long play for both sides is wNc4,Pb3,wPa3,wPc2,bNg8 (NPPPp 15/14), and the more interesting (in my opinion) wKc3,bKe6,bBh1,bNg2,bPe7 (Kkbnp 29/10, Figure 6).

Figure 6: White loses in 29, Black in 10

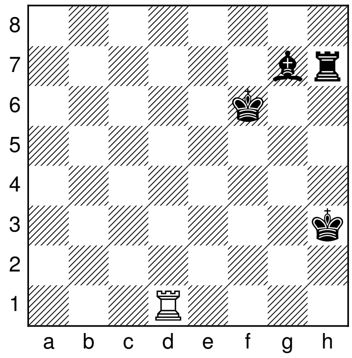


1. Kc2! Nh4!! 2. Kd2 Ng6!! 3. Ke1! Bd5 4. Kf1! Ba2! 5. Kf2! Kf5 6. Ke2! Ke5! 7. Kd1 Bd5! 8. Kc2 Bh1! 9. Kd2! Kd5! 10. Kc2! Nf8! 11. Kb1 Ne6 12. Kc2! Kc5! 13. Kd2! Bd5! 14. Kc1 Kc4 15. Kd1! Kd4! 16. Ke1! Be4! 17. Kf1! Bb1! 18. Kg1 Ke4! 19. Kh2! Nc7 20. Kg1! e5 21. Kf1! Nb5 22. Ke1! Nd6! 23. Kf1! Nc4 24. Kg1! Bc2 25. Kh2! Ke3! 26. Kh3! Bg6! 27. Kh2! Bh5! 28. Kh1! Bg4! 29. Kg1 Kf2! 0-1

1... Kd6! 2. Kb4!! Ke6! 3. Kb5! Ke5! 4. Kb6! Ke4 5. Kc7! e5! 6. Kb6!! Kf5! 7. Kc5! e4! 8. Kb4 e3! 9. Kc3! e2! 10. Kc2!! e1=Q 11. Kd1 1-0

Upon excluding knights/pawns, we have a 13-move loss for White in wKd4,bKg3,bRg6,bBh7,bBh2 (Kkrbb 13/1), and extended losses for both in wRc2,bKe7,bKg4,bRg8,bBf8 (Rkkrb 7/4) and wRd1,bKf6,bKh3,bRh7,bBg7 (Rkkrb 7/4, Figure 7).

Figure 7: White loses in 7, Black in 4



1. Rb1! Ke5!! 2. Rc1! Rh5!! 3. Rb1! Bf6! 4. Rc1! Rh6! 5. Rb1! Kd4! 6. Ra1! Kc3 7. Ra8 Bd8!! 0-1

1... Kh4! 2. Rd2!! Kh5! 3. Rd3! Kh6 4. Rg3 Bh8 5. Rg5 1-0

Overall there are about 195000 full-point zugzwangs, of which about 290 have no Knight or pawn.

APPENDIX B.3: Prior work

Again much of the above was presumably computed in principle (up to rule differences) by Ben Nye about a decade ago, as cited in [B]. Note that this replicates our 78-move KPkrp example (even with rule differences), but indicates that Kknbp tops out at 68 moves (this is a real discrepancy, for the rule differences do not matter).

APPENDIX C: REFERENCES

[B] J. Beasley, *Losing Chess: Burning the Candle at Both Ends*. Variant Chess **41** (Jan 2003), page 8.
`http://www.mayhematics.com/v/vol6/vc41.pdf`