# Chess Revision: Acquiring the Rules of Chess Variants through FOL Theory Revision from Examples

Stephen Muggleton[1], Aline Paes[1,2],
Vítor Santos Costa[3], and Gerson Zaverucha[2]

[1] Department of Computing, Imperial College London, UK
shm@doc.ic.ac.uk
[2] Department of Systems Eng. and Computer Science, UFRJ, Brazil
{ampaes,gerson}@cos.ufrj.br
[3] CRACS and DCC/FCUP, Universidade do Porto, Portugal
vsc@dcc.fc.up.pt

**Abstract.** The game of chess has been a major testbed for research in artificial intelligence, since it requires focus on intelligent reasoning. Particularly, several challenges arise to machine learning systems when inducing a model describing legal moves of the chess, including the collection of the examples, the learning of a model correctly representing the official rules of the game, covering all the branches and restrictions of the correct moves, and the comprehensibility of such a model. Besides, the game of chess has inspired the creation of numerous variants, ranging from faster to more challenging or to regional versions of the game. The question arises if it is possible to take advantage of an initial classifier of chess as a starting point to obtain classifiers for the different variants. We approach this problem as an instance of theory revision from examples. The initial classifier of chess is inspired by a FOL theory approved by a chess expert and the examples are defined as sequences of moves within a game. Starting from a standard revision system, we argue that abduction and negation are also required to best address this problem. Experimental results show the effectiveness of our approach.

## 1 Introduction

Game playing is a fundamental human activity, and has been a major topic of interest in AI communities since the very beginning of the area. Games quite often follow well defined rituals or rules on well defined domains, hence simplifying the task of representing the game as a computer program. On the other hand, good performance in games often requires a significant amount of reasoning, making this area one of the best ways of testing human-like intelligence. Namely, datasets based on games are common testbeds for machine learning systems [7]. Usually, machine learning systems may be required to perform two different kinds of tasks. A first task is to learn a model that can be used to decide whether a move in a game is legal, or not. Having such a model is fundamental

L. De Raedt (Ed.): ILP 2009, LNCS 5989, pp. 123–130, 2010.

for the second task, where one wants to learn a winning strategy. We focus on the the first task in this work.

In order to acquire a meaningful representation of the classifier of the game, one could take advantage of the expressiveness of first-order logic and use *Inductive Logic Programming (ILP)* [11] methods to induce the game's rules written as a logic program, from a set of positive and negative examples and background knowledge (BK). Previous work has demonstrated the feasibility of using ILP to acquire a rule-based description of the rules of chess [8] using structured induction.

Game playing is a dynamic environment where games are always being updated, say, to be more challenging to the player, or to produce an easier and faster variant of the original game. In fact, popular games often have different regional versions. Consider, for example, the game of Chess, arguably, the most widely played board game in the world. It also has been a major game testbed for research on artificial intelligence and it has offered several challenges to the area, since any application involving chess must focus on intelligent reasoning. There are numerous chess variants, where we define *chess variant* as any game that is derived from, related to or inspired by chess, such that the capture of the enemy king is the primary objective [13]. For instance, the *Shogi* game, is the most popular Japanese version of Chess. Although both games have similar rules and goal, they also have essential differences. For example, in Shogi a captured piece may change sides and return to the board [1], which is not allowed in Western Chess.

Knowledge acquisition is a time consuming and error prone task. Ideally, if the rules of a variant of a game have been obtained, we would like to take advantage of them as a starting point to obtain the rules of a variant. However, such rules may need to be modified in order to represent the particular aspects of the variant. In a game such as chess this is a complex task that may require addressing different board sizes, introducing or deleting new promotion and capture rules, and may require redefining the role of specific pieces in the game. Modifying a set of rules so that they could explain a new set of examples is the task of *Theory Revision from Examples* [16]. Besides, transfering a set of rules learnt from a domain to another possibly related domain is studied in the area of *transfer learning* [15,2]. In this work we handle the problem of transfering the rules of chess to some variants of this game as an instance of Theory Revision from Examples. This task is particularly challenging due to the necessity of representing all the official rules of the game, which makes the theory be very large, with several branches and sub-concepts being used to classify a correct move.

We show that we can learn rules between different variants of the game of chess. Starting from the state-of-the-art FORTE revision system [14,5], we contribute with **(i)** a new strategy designed to *simplify* the initial theory by removing rules that will *not* be transferred between variants; **(ii)** support for *abduction*; and **(iii)** support for *negation as failure*. Experimental evaluation on real variants of chess

[1] It is suggested that this innovative drop rule was inspired by the practice of 16th century mercenaries who switched loyalties when captured [13].

shows that our technique can transfer between variants with smaller and larger boards, acquire unusual rules, and acquire different pieces.

The paper is organized as follows. We start by discussing the revision system, emphasizing the modifications performed on it in section 2. Next, we briefly describe the examples and the initial knowledge in section 3. We present experimental results in section 4 and finally we conclude the paper and discuss future work in section 5.

## 2 Chess Revision

Inductive Logic Programming (ILP) systems learn using a set of examples and background knowledge (BK), both described as logic programs and assumed as correct. On the other hand, theory revision from examples consider that the BK could also contain incorrect rules, which, after being modified, would better reflect the dataset. Thus, in theory revision the BK is divided into two sets: one containing the rules subject to modification, named here as *initial theory* and the other composed of rules known to be correct and therefore not modifiable, containing intensional definitions of the fundamental relations used to define a domain, named as *fundamental domain theory (FDT)* [14]. The goal of theory revision is to identify points in the initial theory preventing it from correctly classifying some example, and propose modifications to such points, so that the revised theory together with the FDT is correct. Theory Revision is particularly powerful and challenging because it must deal with the issues arising from revising multiple clauses (theory) and even multiple predicates (target).

Usually, the first step in a revision process is to find the clauses and/or antecedents responsible for the misclassification of some example, the *revision points*. Then, modifications are suggested to these revision points, through *revision operators*, such as rule deletion and antecedent addition, if one wants to specialize, or antecedent deletion or rule addition, if one wants to generalize [16].

In this work we follow the new version of FORTE revision system [14] as modified in [5] to allow the use of bottom clause and modes declarations [9] when searching for literals to be added to the body of clauses. In order to best address the revision of the rules of Chess, we performed three main modifications on the current version of the system, described as follows.

*Starting the revision process by deletion of rules.* In an attempt to decrease the complexity of the theory and consequently of the whole revision process, we introduced a first step of deletion of rules. This process is performed as a hill-climbing iterative procedure, where at each iteration the facts used in proofs of negative examples are selected, the deletion of each one is scored and the one able of improving the score at most is chosen. This step restarts from the modified theory and finishes when no deletion is able to improve the score. We found this procedure both reduces theory size and noise, namely when the target theory is a specialized version of the initial theory.

*Using abduction during the revision process.* Abduction is concerned about finding explanations for observed facts, viewed as missing premises in an argument, from available knowledge deriving those facts [6]. Usually, theory revision systems, including FORTE, use abduction when searching for generalization revision points, to locate faults in a theory and suggest repairs to it, determining a set of assumptions that would allow the positive example be proved [14]. We further benefit from abduction in two distinguished moments of the revision process. First, when searching for generalization revision points (points failing on proving positive examples), we assume that faulting abducible predicates are true and continue to search for further revision points possibly depending on them. Note that in this moment the literal is ground. Then, such abducible predicates are included in the theory under revision, up to a maximum number of included abducible predicates. The abducible predicates might be eventually generalized/specialized in the next iterations, in case they become a faulting point in the theory.

The second further moment we use abdcution is when constructing a bottom clause for intermediate predicates. Those are predicates in the head of clauses but the dataset does not contain examples for them, since the examples are of the top-level predicate(s) only. However, to construct the bottom clause it is necessary to start from a positive example with the same predicate as the head of the clause being specialized. Thus, from a positive example proved by the current clause we obtain the required literal using the initial theory and FDT. The procedure takes a top-level goal (an example) and the intermediate predicate and instantiates such a predicate to the first call to that predicate encountered when attempting to prove the goal. The proof starts from the example and return the instantiation of the specified intermediate predicate. Next, we construct the bottom clause from such a literal and use it as search space for adding literals to the clause. In [10] a similar procedure is used to obtain "contra-positive examples" to the intermediate predicates and then construct a bottom clause for them.

*Using negated literals in the theory.* FORTE was neither able to introduce negated literals in the body of the clause nor revise negated antecedents. Negation is essential to elegantly model the chess problem, since we need to represent concepts such as *the king is* **not** *in check*, among others. In order to add negated literals in the body of the clause, we allow the bottom clause procedure construction to generate negated literals. To do so, one could take advantage of constructive negation techniques [3,4], ensuring that there are no free variables in any `not`(*Goal*) that might be called. We also introduced a procedure for handling a faulty negated literal during the revision process. Roughly speaking, if the negated literal is responsible for a failed proof of positive examples, it is treated as a specialization revision point. On the other hand, if the negated literal takes part in a proof of a negative example, it is treated as a generalization revision point. This is a preliminary attempt at introducing non-monotonic reasoning in FORTE.

## 3 The Examples and the Background Knowledge

*Examples.* In our framework, the legal moves in the chess variant are the positive examples and the illegal moves are the negative examples. The dataset is composed of a set of simulated games, where each game is limited to a specified maximum number of legal moves, considering both players. The moves are within a game, aiming to represent castling and en-passant, which require the history of the game (p.ex., the rook and the king involved in a castling must have not moved yet in the whole game), and promotion, which requires updating the board to represent the promoted piece. We take advantage of FORTE examples representation where the examples are in the format

*Ground instances of target predicate* ← *Conjunction of facts from the context*

and we represent each game as a set of legal and illegal moves within the game (they are the ground instances of the target predicate) and the positions of the pieces related to each round (the board of the game) are the facts from the context. Thus, each game has its separate set of legal and illegal moves and set of positions of the pieces during the game, in the format

$Target\ Predicate$ :
$Positives$ :
$move(Round, Piece_1, Colour_1, File_1, Rank_1, Piece_2, Colour_2, File_2, Rank_2), ...$
$Negatives$ :
$move(Round, Piece_1, Colour_1, File_1, Rank_1, Piece_2, Colour_2, File_2, Rank_2), ...$
$Context$ :
$board(Round, Piece, Colour, File, Rank), ...$
$out\_board(Round, Piece, Colour, –1, –1), ..., out\_board(Round, Piece, Colour, 0, 0), ...$

The terms of the ground instances of the target predicate are the number of the move, the current and next status of the piece. For example, $move(9, pawn, white, c, 7, rook, white, c, 8)$ states that in the $9th$ move executed in the game a *white pawn* moves from $c, 7$ to $c, 8$ and is promoted to a *rook*. The facts from the context represent the position of the pieces on the board and the pieces removed from the games. Considering the example above, the set of facts would contain $board(10, rook, white, c, 8)$, stating that after the $9th$ move, there is a *white rook* in the position $c, 8$ on the board and $out\_board(10, pawn, white, 0, 0)$, giving the information that a *white pawn* was removed from the board. The board setting is updated according to the legal move(s) performed on the previous round. A move generator procedure is responsible for creating the dataset of simulated games.

*Background Knowledge.* In the chess revision problem, the initial theory describes the rules of the standard game of chess, which will be revised using the set of examples for its variant. The theory encompass all the possible branches of a move, considering the official rules of chess [1]. For example, *in case* the king is under attack, a piece *must* only move to stop the attack, by *either* capturing the attacking piece *or* moving to the way between the king and the attacking piece. This theory is inspired on the one learned in [8] using hierarchical structured induction

ILP and Progol [9]. There, the BK was composed of 61 clauses and 61 clauses were learned.[2] The major differences between the theory used in the present work and the previous one are (1) the clauses we include to describe castling, en-passant and promotion, since the authors of that work opted to not represent any such special move and (2) the clauses we modified to use constructive negation, to avoid free variables in negated literals.

The FDT contains fundamental rules to the problem of chess in general, such as to calculate the difference between two positions (files or ranks) of pieces on the board, definitions of equality, among others. This knowledge is valid to the standard chess and also to the chess variants. During the revision process, the clauses in FDT are not modified. It is composed of 40 clauses, approximately.

The initial theory describes the rules for achieving a legal move following the rules of chess. The top-level predicate is the same as the target predicate, related to the move of the piece. The sub-concepts such as *simple and double check*, *pin*, *attack*, *promotion* and *basic moves of each piece* are described through intermediate predicates. The pieces, files, ranks and colours used in the game are enumerated by facts. All in all, the initial theory has approximately 100 clauses.

## 4 Experimental Results

*Experimental methodology.* To experiment the proposal of the paper, we generated datasets with 5 simulated games where each stage of the game has 1 positive and 5 negative examples and the maximum number of legal moves is 20, for 3 different chess variants. We performed 5-fold cross validation and scored the revisions using f-measure. We proceed with this section by describing the chess variant followed by the automatic revisions performed by the system to obtain its theory. The variants include a smaller version of chess, a version with an unusual rule and a variant with larger board and unusual pieces.

- Using smaller boards: Gardner's Minichess: This is the smallest chess game (5X5) in which all original chess pieces and legal moves are still used, including pawn double move, castling and en-passant [13]. The revisions are as follows.
  1. The delete rule step was able to remove the following clauses from the theory: file(f). file(g). file(h). rank(6). rank(7). rank(8). promotion_zone (pawn,white,7,8).
  2. The add rule generalization operator created the following clause: promotion_zone(pawn,white,4,5) (white pawn promoting in rank 5).

  The final accuracy was 100% and the returned theory matches the rules of Gardner Minichess.
- Unusual rule: Free-capture chess: This variant of chess allows a piece to capture friendly pieces, except for the friendly king, besides the opponent pieces [13].

[2] The resulting theory was approved by Professor Donald Michie, who could be considered a chess expert.

1. The chess theory makes sure that a piece when attacking does not land on a friendly piece, requiring that the colours of them both are different. The revision created a new clause by first deleting the predicate requiring the colours are different and then adding a literal restricting the attacked pieces to be different from the king. Note that the original rule is kept on the theory.
2. There is a specific rule in the case of a king attacking, since the restrictions on the king's move must be observed. The revision system deleted the literal requiring the colour of a piece attacked by the king be different from the colour of the king.

The final accuracy was 100% and we can say that the returned theory perfectly corresponds to the target theory.

– Unusual pieces and larger board: Neunerschach: This is a chess variant played on a 9x9 board. There is a piece called as *marschall* replacing the queen and moving like it. The extra piece is the *hausfrau*, which moves as a queen but only two squares [13]. The theory was revised as follows.

1. The delete rule step removed the clause defining the queen as a piece;
2. The abduction procedure included facts defining the $marschall$ and $hausfrau$ as pieces on the theory;
3. From the rule defining the basic move of the queen, the add rule operator created a rule for the $marschall$.
4. New rules were added to the theory, defining the basic move of $hausfrau$ and introducing the facts $file(i)$ and $rank(9)$.

Since in this dataset no promotion moves were generated, due to the size of the board, the revision process failed on correcting the promotion on the last rank. We expect that using games with a larger number of total moves will allow us to represent such a promotion. Nevertheless, the final accuracy was 100%.

## 5 Conclusions

We presented a framework for applying the knowledge learned to the rules of chess to learn variants of chess through theory revision and a set of generated examples. We described the modifications implemented in the revision system, including the introduction of an initial step for deleting rules, the use of abduction and negation. Three variants of chess were experimented and the system was able to return final theories correctly describing the rules of the variants, except for one case, the promotion in the last rank of a 9X9 board.

In order to decrease the runtime of the revision process we intend to use the stochastic local search algorithms developed in [12]. We would like to try induce the variants of Chess using standard ILP system, such as Progol [9], with the chess theory as BK. Beforehand, it is expected that these systems are not successful in the cases requiring specialization, since they do not perform such operation. If the chess theory is not used as BK, we would not take advantage

of the initial knowledge about the domain. Additionally, we want to take a further step towards the acquirement of more complex chess variants, such as the regional chess games *Shogi* and *Xiangqi*.

## Acknowledgements

Stephen Muggleton would like to thank the Royal Academy of Engineering and Microsoft Research for funding his Research Chair in Machine Learning. Aline Paes and Gerson Zaverucha are financially supported by Brazilian Research Council (CNPq) and Vítor Santos Costa by Fundação para a Ciência e Tecnologia.

## References

1. Burg, D.B., Just, T.: IUS Chess Federation Official Rules of Chess. McKay, New York (1987)
2. Caruana, R.: Multitask Learning. Machine Learning 28(1), 41–75 (1997)
3. Chan, D.: Constructive Negation Based on the Completed Database. In: Proc. of the 5th Int. Conf. and Symp. on Logic Programming, pp. 111–125. The MIT Press, Cambridge (1988)
4. Drabent, W.: What is Failure? An Approach to Constructive Negation. Acta Inf. 32(1), 27–29 (1995)
5. Duboc, A.L., Paes, A., Zaverucha, G.: Using the Bottom Clause and Mode Declarations in FOL Theory Revision from Examples. Machine Learning 76, 73–107 (2009)
6. Flach, P., Kakas, A.: Abduction and Induction: Essays on their Relation and Integration. Kluwer Academic Publishers, Dordrecht (2000)
7. Fürnkranz, J.: Recent Advances in Machine Learning and Game Playing. OGAI-Journal 26(2), 147–161 (2007)
8. Goodacre, J.: Master thesis, Inductive Learning of Chess Rules Using Progol. Programming Research Group, Oxford University (1996)
9. Muggleton, S.: Inverse Entailment and Progol. New Generation Computing 13(3&4), 245–286 (1995)
10. Muggleton, S., Bryant, C.H.: Theory completion using inverse entailment. In: Cussens, J., Frisch, A.M. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 130–146. Springer, Heidelberg (2000)
11. Muggleton, S., De Raedt, L.: Inductive Logic Programming: Theory and Methods. J. Log. Program. 19/20, 629–679 (1994)
12. Paes, A., Zaverucha, G., Costa, V.S.: Revising FOL Theories from Examples through Stochastic Local Search. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 200–210. Springer, Heidelberg (2007)
13. Pritchard, D.B.: The Classified Encyclopedia of Chess Variants. John Beasley (2007)
14. Richards, B.L., Mooney, R.J.: Automated Refinement of First-order Horn-Clause Domain Theories. Machine Learning 19(2), 95–131 (1995)
15. Thrun, S.: Is Learning the nth Thing any Easier than Learning the First? In: Adv. in Neural Inf. Proc. Systems. NIPS, vol. 8, pp. 640–646. MIT Press, Cambridge (1995)
16. Wrobel, S.: First-order theory refinement. In: De Raedt, L. (ed.) Advances in Inductive Logic Programming, pp. 14–33. IOS Press, Amsterdam (1996)