QBF ソルバを用いた将棋に対する即詰みの有無判定

石脇　滉己[1]　吉仲　亮[1]　篠原　歩[1]

概要：本研究の目標は，与えられた将棋の局面が入力された手数以内に詰むかどうかの問題を True Quantified Boolean Formula (TQBF) 問題に帰着して，QBF ソルバを用いることで即詰みの有無判定を行う手法を確立することである．TQBF 問題とは，与えられた Quantified Boolean Formula (QBF) が充足可能であるかを判定する問題である．この問題は，PSPACE 完全問題として知られており，理論的には高速に解くことは非常に困難である．これに対して，この TQBF 問題を実用的に高速に解くために QBF ソルバが開発されてきた．本稿では，持ち駒のない局面を QBF に帰着する方法を提案し，簡単な局面に対しては即詰みの有無判定を正しく行えることを確認した．

Determining Forced Mate Possibility in Shogi using a QBF Solver

Koki Ishiwaki[1]　Ryo Yoshinaka[1]　Ayumi Shinohara[1]

Abstract: The goal of this research is to establish a method to solve the problem to decide whether there is a forced mate within inputted number of moves on a given shogi position by reducing it to the true quantified boolean formula (TQBF) problem and then using a QBF solver. The TQBF problem is to determine whether a given quantified boolean formula (QBF) can be satisfied. This problem is known as PSPACE complete, so it is very difficult to solve fast in theory. In order to solve the TQBF problem fast in practice, various QBF solvers have been developed. In this paper, we propose a method to reduce positions without pieces in hand to QBF and verified that a QBF solver can determine the forced mate possibility of simple position.

1. はじめに

近年，将棋や囲碁においてはプロレベルの AI が実現されている．これらの AI を支えている技術として α-β 法や証明数探索 [1] などといった手法がある．

それに対して，本研究ではこれらの既存の探索手法ではなく QBF ソルバを用いて，与えられた将棋の局面に即詰みが存在するかどうかの判定を行う．QBF ソルバとは True Quantified Boolean Formula (TQBF) 問題を解くプログラムのことである．また，即詰みとは王手の連続でそのまま詰みに至ることである．

2 人完全情報ゲームの勝敗判定問題を TQBF 問題に帰着して，QBF ソルバを用いて勝敗判定を行う研究としては [2] [3] [4] がある．ここで，ゲームの勝敗判定問題とは，お互いのプレイヤーが最善手を選択し続けた場合，どちらのプレイヤーが勝利するのか，あるいは引き分けになるのかを判定する問題である．

本研究では [2] [3] [4] に倣い，与えられた将棋の局面が入力された手数以内に詰むかどうかの問題を TQBF 問題に帰着して，QBF ソルバを用いて即詰みの有無判定を行う手法を提案する．これまで将棋に対して汎用の QBF ソルバを用いる研究は我々の知る限り存在しないが，(1) 将棋に対して QBF ソルバがどの程度有効であるのか確認する，(2) QBF ソルバの得意な問題あるいは苦手な問題を探りソルバの性能を向上させる，という動機から本研究では，QBF ソルバを用いて即詰みの有無判定を行う．

本稿では，実現へ向けての第一歩として，持ち駒の無い局面に対する即詰みの有無判定を行い，提案手法の有用性について検討を行った．

[1] 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University

```
1 p cnf 3 2
2 a 1 0
3 e 2 0
4 a 3 0
5 1 -2 0
6 3 0
```

図 1　式 (3) を QDIMACS 形式で表した例

2. TQBF 問題

2.1 QBF

Quantified Boolean Formula (QBF) とは，次のように表される論理式のことである．

$$Q_1x_1Q_2x_2...Q_nx_n.\phi(x_1,x_2,...,x_n) \quad (1)$$

ここで，各 Q_i $(i = 1,2,\ldots,n)$ は $\forall$ または $\exists$ であり，$\phi(x_1,x_2,\ldots,x_n)$ は $x_1,x_2,\ldots,x_n$ を変数としたブール式である．

QBF の例を式 (2) と式 (3) に示す．

$$\forall x_1\exists x_2.(x_1 \vee \neg x_2) \quad (2)$$

$$\forall x_1\exists x_2\forall x_3.((x_1 \vee \neg x_2) \wedge x_3) \quad (3)$$

式が恒真ならば充足可能，恒真でないならば充足不可能となるので，式 (2) は充足可能 (SAT)，式 (3) は充足不可能 (UNSAT) となる．このように，QBF が充足可能であるかを判定する問題を TQBF 問題という．

2.2 QBF ソルバ

TQBF 問題は PSPACE 完全問題として知られており，理論的には高速に解くことは非常に困難であるが，実用的に高速に解くアルゴリズムが開発されてきた．これまで開発されてきた QBF ソルバとしては DepQBF [5] や RAReQS [6] などが挙げられる．本稿では，既存の QBF ソルバである DepQBF を用いて局面に対して詰み手順が存在するかどうかの判定を行う．

DepQBF の入力として与える形式は QDIMACS 形式である [7]．QDIMACS 形式とは，QBF ソルバの入力および出力の標準フォーマットの一つである．なお，QDIMACS 形式では式 (1) の $\phi(x_1,x_2,...,x_n)$ の部分は CNF で記述する必要がある．

例として，式 (3) を QDIMACS 形式で表したものを図 1 に示す．1 行目の `p` と `cnf` に続く数字はそれぞれ変数の数と節の数を表している．つまり，図 1 で使用する変数の数は 3 つ，節の数が 2 つということである．2〜4 行目にある `a` と `e` は，`a` が $\forall$ を，`e` が $\exists$ を表している．また，`1`〜`3` の数字は変数を番号で表したもので，式 (3) の変数 x_1〜x_3 がそれぞれ `1`〜`3` に対応している．そして，`0` は行の終端を表す．5〜6 行目は節について書かれており，CNF に対応させると空白で OR，マイナスで NOT を表している．

3. 提案手法

本研究では，QBF ソルバを用いて，与えられた局面に即詰みが存在するかどうかの判定を行う．

3.1 手法の概要

QBF ソルバを用いて，局面に即詰みが存在するかどうかの判定を行う際の全体の流れを図 2 に示す．

初めに，QBF を生成するプログラムに，局面の情報（駒の位置，盤面の大きさ）と手数を入力し，それに応じた QBF を生成する．その後，生成された QBF を QBF ソルバに入力することで，与えられた局面に対して入力された手数以内に即詰みがあるか（SAT），あるいは存在しないか（UNSAT）を出力する．

3.2 QBF エンコーダ

与えられた局面において入力された手数以内で即詰みがあるかの判定を QBF ソルバで行うために，QBF を生成する必要がある．本研究では，この QBF を生成するプログラムを QBF エンコーダと呼んでいる．この QBF エンコーダで出力される QBF の構成は次のようになっている．

$\exists$(初期状態に関する変数).

$\exists$(先手 1 手目を表す変数).$\exists$(1 手目の状態に関する変数).

$\forall$(後手 2 手目を表す変数).$\exists$(2 手目の状態に関する変数).

$\vdots$

$\exists$(n 手目を表す変数).$\exists$(n 手目の状態に関する変数).

$\forall$($n+1$ 手目を表す変数).$\exists$($n+1$ 手目の状態に関する変数).

ϕ(上記の変数すべて)

ここで，m 手目 $(m = 1,2,...,n+1)$ の状態とは m 手目を指した後の局面やその局面が詰みか不詰みかなどのゲームの状態のことを指し，n は入力された手数を指す．$n+1$ 手目まで後手の手がある理由としては，先手の指し手に対してどのように応じても王手から逃れることができないことを確かめるためである．また，ϕ は詰みの判定や指し手のルールなどを書いたブール式で構成されている．

今回のプログラムで実装したブール式 ϕ については多くのブール式で構成されているため，すべてを本稿で説明することはできない．そのため，本稿では例として実装した一部である「王手判定に関するブール式」について説明する．実装した王手判定に関するブール式を式 (4) に示す．また，王手がかかっている局面の例を図 3 に示す．

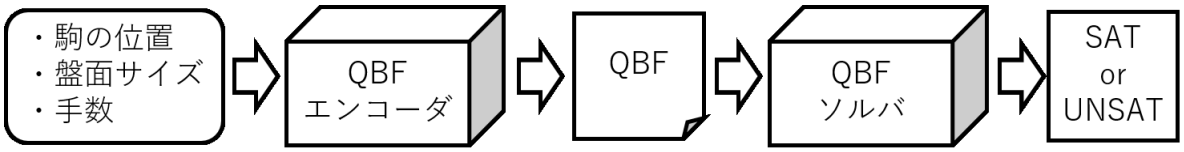

図 2 局面に詰みが存在するかどうかの判定を行う流れ

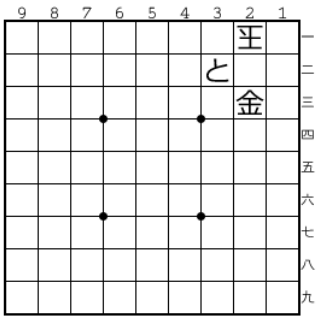

図 3 王手がかかっている局面[*1]

$$\bigwedge_{z=1}^{Z}\left(\bigvee_{x=1}^{W}\bigvee_{y=1}^{H}(gote\_ou_{z,x,y} \wedge sente\_kiki_{z,x,y}) \Leftrightarrow check_z\right) \tag{4}$$

式 (4) で使用している変数について説明する．まず，W と H はそれぞれ盤面の幅と高さを表している．また，$gote\_ou_{z,x,y}$ は z 手目に後手玉が地点 (x, y) に存在する場合に真になる変数であり，$sente\_kiki_{z,x,y}$ は z 手目に先手の利きが地点 (x, y) にある場合に真になる変数である．つまり，図 3 の場合では $gote\_ou_{z,x,y}$ と $sente\_kiki_{z,x,y}$ の値は図 4 のようになる．図 4 に $gote\_ou_{z,x,y}$ と $sente\_kiki_{z,x,y}$ の値を示す．そして，$check_z$ は z 手目に後手玉に対して王手がかかっている場合に真になる変数である．この例の場合では，式 (4) の $gote\_ou_{z,x,y}$ と $sente\_kiki_{z,x,y}$ の値が図 4 のようになるので，$check_z$ の値は真になる．つまり，この局面は王手がかかっているという判定になる．

この他にも，「駒の動きに関するブール式」や「成に関するブール式」，「合法手であるか確認するブール式」なども実装しており，本研究ではこれらのブール式によって与えられた局面に対して即詰みの有無判定を行っている．

なお，QBF エンコーダは開発途上であり，現在のところ持ち駒には対応できていない．

3.3 先行研究との違い

先行研究 [2] では，Connect-4 を一般化した「Connect-c」に対して， [3] [4] では一般化三並べを拡張したゲームである「GTTT(p, q)」に対して，それぞれ，QBF ソルバを用いてゲームの勝敗判定を行っていた．本研究ではこれらの先行研究を参考にしているがいくつか異なる点がある．

まず，1 つ目としては本研究では局面の情報だけでなく，手数も入力する点である．先行研究で扱っていた Connect-c や GTTT(p, q) では一度置いた石はゲームが終わるまで動くことはないため，盤面サイズからゲームの最大手数を求めることができた．それに対して，将棋は現在の局面から何手でゲームが終わるのかが求められない場合がほとんどである．そこで，本研究では最大手数を入力として与えることにした．

2 つ目の点としては，将棋では盤面上の駒が移動する点である．Connect-c や GTTT(p, q) べでは，石が置かれていない位置さえ指定すれば着手することが可能である．しかし，将棋においてはどの駒がどこからどこへ移動するのか分からなければ着手することができない．さらに，駒によって動き方が異なるため，先行研究に比べてより盤面の管理が難しくなっている．

[*1] 局面図の作成には shogipic.jp を使用した

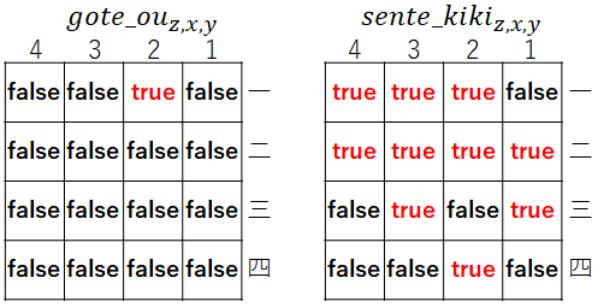

図 4 図 3（1 一の地点から 4 四の地点まで）の場合における $gote\_ou_{z,x,y}, sente\_kiki_{z,x,y}$ の値

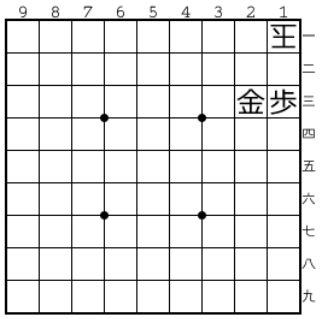

図 5 1 手詰の問題

4. 実験結果

今回実装したプログラムを用いて，与えられた局面に対して即詰みの判定が正しく行えるか確認を行う．図 5 に 1 手詰の問題，図 6 に 3 手詰の問題を示す．

図 5 の問題は ▲1 二金あるいは ▲1 二歩成の 1 手詰であり，図 6 の問題は ▲3 二歩成 △1 一玉 ▲2 二金（あるいは

表 1 図 5，図 6 に対する実行結果

問題	盤面サイズ	手数	変数	節	結果	実行時間（秒）	
						QBF エンコーダ	QBF ソルバ
図 5	3 × 3	1 手	3860	14119	SAT	0.888	0.203
図 6	3 × 3	1 手	3860	14119	UNSAT	0.979	0.203
		3 手	6880	25051	SAT	1.842	2.375

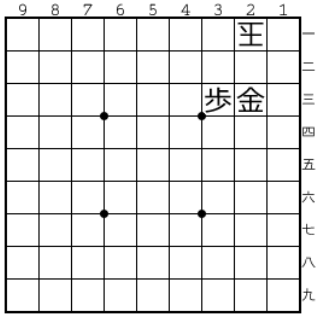

図 6 3 手詰の問題

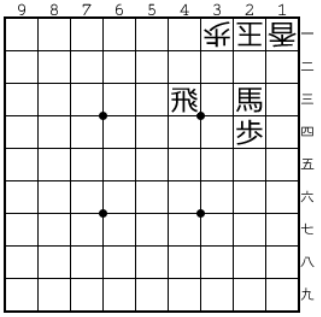

図 7 詰将棋集の 3 手詰問題 (1)

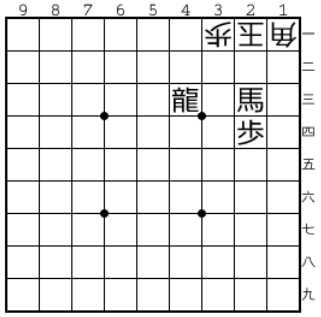

図 8 詰将棋集の 3 手詰問題 (2)

▲2 二と）までの 3 手詰である．

実際に，図 5 と図 6 に対して本システムを用いた結果を表 1 に示す．なお，通常の将棋では敵陣は 3 段目までであるが，図 5 と図 6 の問題では敵陣は 2 段目までとしている．つまり，先手の駒が 1,2 段目に侵入した場合に成ることができる．また，盤面のサイズは 3 × 3（1 一の地点から 3 三の地点まで）として，QBF ソルバは DepQBF 6.0 [5] を使用した．表 1 に示すように，図 5，図 6 ともに正しい判定が返せていることが分かる．また，図 5 の問題よりも図 6 のほうが QBF ソルバの実行時間が増えていることから，手数が大きくなるにつれて QBF ソルバの実行時間も増えていくと考えられる．

次に，市販の詰将棋集 [8] に載っている作品に対して即詰み（SAT）であると判定を返すことができるか確認をする．図 7 と図 8 に詰将棋集の 3 手詰問題を示す．なお，図 7 と図 8 の問題は持ち駒を必要としない問題である．図 7 の問題は ▲2 二馬 ▲△ 同玉 ▲2 三飛成までの 3 手詰であり，図 8 の問題も ▲1 二馬 △ 同玉 ▲2 三龍までの 3 手詰である．

実際に，図 7 と図 8 に対して本システムを用いた結果を表 2 に示す．なお，図 7 と図 8 の問題では敵陣は通常の将棋と同じく 3 段目までとし，盤面のサイズは 4 × 4（1 一の地点から 4 四の地点まで）とした．表 2 に示すように，図 7，図 8 ともに正しい判定が返せていることが分かる．しかし，表 1 の結果と比較すると，QBF ソルバの実行時間が増えていることが分かる．このことから，盤面サイズが大きくなるにつれて実行時間が増えていくと考えられる．

これらの結果から，盤面サイズと手数が大きくなるにつれて QBF ソルバの実行時間が増えていくと考えられる．そこで，実際に盤面サイズと手数が大きくなるとどのように実行時間が増えていくか検証する．図 5 の問題に対して，盤面サイズを 3 × 3 で固定して手数を変更した場合の結果を表 3 に示す．表 3 に示すように手数が大きくなるにつれて実行時間が少しずつ増加していることが確認できる．しかし，実行時間は増えているが実用的な時間で求められていることが分かる．

また，図 5 の問題に対して，手数は 1 手で固定して盤面サイズを変更した場合の結果を表 4 に示す．ここで $n \times n$ の盤面とは，1 一の地点を 1 つの頂点とした 1 辺 n の正方盤面のことである．表 4 をみてみると盤面サイズが大きくなるにつれて実行時間が増加していることが分かる．通常の将棋の盤面サイズでも実用的な時間で詰将棋を解けるようにするためにも，この問題を解決していく必要があると考えられる．

次に，図 5 の問題に対して関係のない変数を削除して，手数と盤面サイズをそれぞれ大きくしたときの実行時間を確認する．表 5 に盤面サイズを 3 × 3 で固定して手数を変更した場合の結果を，表 6 に手数を 1 手で固定して盤面サイズを変更した場合の結果を示す．ここで，図 5 の問題に対して関係のない変数とは「歩，金，玉，と金」以外の駒に関する変数（使用しない駒に関する変数）のことである．

表 2 図 7，図 8 に対する実行結果

問題	盤面サイズ	手数	変数	節	結果	実行時間（秒）	
						QBF エンコーダ	QBF ソルバ
図 7	4 × 4	1 手	7227	30954	UNSAT	2.105	3.000
		3 手	12963	54570	SAT	3.758	28.531
図 8	4 × 4	1 手	7227	30954	UNSAT	2.128	5.891
		3 手	12963	54570	SAT	3.883	17.328

表 3 図 5 の問題に対して手数を変更した場合

盤面サイズ	手数	変数	節	結果	実行時間（秒）	
					QBF エンコーダ	QBF ソルバ
3 × 3	1 手	3860	14119	SAT	0.888	0.203
	3 手	6880	25051	SAT	1.767	0.922
	5 手	9900	35983	SAT	2.536	2.422
	7 手	12920	46915	SAT	3.357	3.547
	9 手	15940	57843	SAT	4.240	4.844
	11 手	18960	68779	SAT	4.978	6.078

表 4 図 5 の問題に対して盤面サイズを変更した場合

手数	盤面サイズ	変数	節	結果	実行時間（秒）	
					QBF エンコーダ	QBF ソルバ
1 手	3 × 3	3860	14119	SAT	0.888	0.203
	4 × 4	7227	30954	SAT	1.877	1.375
	5 × 5	12132	59907	SAT	3.429	7.750
	6 × 6	19007	106594	SAT	6.134	36.922
	7 × 7	28380	177975	SAT	10.088	169.938
	8 × 8	40875	282354	SAT	15.419	539.219
	9 × 9	57212	429379	SAT	23.775	1714.484

表 5 変数を削除して図 5 の問題に対して手数を変更した場合

盤面サイズ	手数	変数	節	結果	実行時間（秒）	
					QBF エンコーダ	QBF ソルバ
3 × 3	1 手	1316	3560	SAT	0.242	0.047
	3 手	2440	6586	SAT	0.439	0.141
	5 手	3564	9612	SAT	0.636	0.391
	7 手	4688	12638	SAT	0.848	0.594
	9 手	5312	15664	SAT	1.023	0.969
	11 手	6936	18690	SAT	1.258	1.281

表 6 変数を削除して図 5 の問題に対して盤面サイズを変更した場合

手数	盤面サイズ	変数	節	結果	実行時間（秒）	
					QBF エンコーダ	QBF ソルバ
1 手	3 × 3	1316	3560	SAT	0.242	0.047
	4 × 4	2751	7351	SAT	0.496	0.281
	5 × 5	5172	13420	SAT	0.908	2.469
	6 × 6	9011	22631	SAT	1.502	16.906
	7 × 7	14796	36040	SAT	2.390	79.375
	8 × 8	23151	54895	SAT	3.657	380.188
	9 × 9	34796	80636	SAT	5.464	1247.344

変数を削除した理由としては，問題とは関係ない駒の変数が実行時間にどれほど影響しているのか確認するためである．表 5 と表 3 の結果を比較すると，QBF ソルバの実行時間が短くなっていることが分かる．また，表 6 と表 4 の結果を比較した場合でも QBF ソルバの実行時間が短くなっている．これらの結果から，問題で使用していない駒の変

数が QBF ソルバの実行時間にある程度影響を及ぼしていることが分かる．しかしながら，削除された変数と節の数から考えると，使用していない駒の変数はそれほど実行時間に影響していないと考えられる．

5. まとめ

本稿では与えられた局面に対して即詰みが存在するかどうかの判定を正しく行えていることを確認した．しかしながら，現在のところ，持ち駒に関しては未実装である．今後は持ち駒を導入して，どのような問題に対しても本システムが使用できるようにする予定である．

表 4 の結果から分かるように，盤面のサイズが大きくなるほど実行時間も増加して実用的な時間で判定を返すことができていないので，将来的にはこの問題を解決していく必要あると考えられる．

また，本稿では QBF ソルバとして DepQBF 6.0 のみを使用したが，今後は他の QBF ソルバも使用してそれぞれの QBF ソルバの性質についても確認していきたい．

参考文献

[1] L.Victor Allis, Maarten van der Meulen, and H.Jaap van den Herik. Proof-number Search. *Artificial Intelligence*, Vol. 66, No. 1, pp. 91–124, 1994.

[2] Ian P Gent and Andrew G D Rowley. Encoding Connect-4 using Quantified Boolean Formulae. *2nd Intl. Work. Modelling and Reform. CSP*, pp. 78–93, 2003.

[3] Diptarama, Ryo Yoshinaka, and Ayumi Shinohara. QBF Encoding of Generalized Tic-Tac-Toe. *Proceedings of the 4th International Workshop on Quantified Boolean Formulas*, pp. 14–26, 2016.

[4] ディプタラマ, 石黒裕也, 成澤和志, 篠原歩, ジョーダン・チャールズ. QBF ソルバを用いた一般化三並べの拡張の勝敗判定. ゲームプログラミングワークショップ 2015 論文集, pp. 154–161, 2015.

[5] Florian Lonsing and Uwe Egly. DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL. *Proceedings of the 26th International Conference on Automated Deduction*, pp. 371–384, 2017.

[6] Mikoláš Janota, William Klieber, Joao Marques-Silva, and Edmund Clarke. Solving QBF with Counterexample Guided Refinement. *Theory and Applications of Satisfiability Testing–SAT 2012*, pp. 114–128, 2012.

[7] M.Narizzano and A.Tacchella. Qdimacs prenex cnf standard ver. 1.1(2005). `http://www.qbflib.org/qdimacs.html`.

[8] 森信雄. 詰将棋ドリル 3　3 手詰初級編. 株式会社廣済堂, 2012.