System Reduction Based on Symmetry in Game Model Checking

Lin-feng JIAO* and Qing-lei ZHOU
Zhengzhou, China
*Corresponding author

Keywords: Symbolic model checking, Symmetry, Game verification, Alternating-time temporal logic, Go problem.

Abstract. In the research of two-player games, verifying whether there exists winning strategy is a major problem of game theory. Based on the high efficiency of symbolic model checking, model checking can be applied to game verification. However, some games like game of Go have large state space, so reduction of system model is a key issue to improve the scale and efficiency of verification. This paper presents a method of reducing system model of Alternating-time Temporal Logic (ATL) model checking based on symmetry and gives two experiments on tic-tac-toe and go in small board.

Introduction

Zero sum game in game theory has been a research hotspot in the field of artificial intelligence. In computer game research, some researchers focus on enhancing computer's ability of playing chess by pruning search and probability calculation, so that computer can defeat human player. In addition, some researchers try to find and verify a winning strategy of a game, so that no matter how the opponent moves, a player can always win. If a winning strategy exists under any situation, the game is strongly solvable. Then a player can always win the game as long as he moves by the winning strategy. However, for some games like Go, the state space is too large to find winning strategy.

In March of 2016, the Go program “Alpha-Go” defeated the professional 9 dan player Lee SeDol and won the whole world’s attention. According to [1], “Alpha-Go” based on probability and random computation, does not guarantee that every move takes the best strategy. With a great number of matches of human experts and self play, “Alpha-Go” program chooses strategies that most likely to win instead of searching entire state space.

In the game tree, pruning some branches which are almost impossible to win can bring much efficiency. However, some information may loss during pruning. In some situations, loss of information caused by non-deterministic mechanism is unacceptable. Black 79 of the fourth match between Lee SeDol and Alpha-Go is an example, which directly causes a failure of Alpha-Go. Moreover, it is possible to search the state space in a certain range, while the probability and random computation are not entirely reliable. According to this, we need a more determined approach, by which we can achieve a balance between efficiency and information integrity.

Reference [2] has introduced model checking [3] to search the state space in a certain range for Go. However, the state space of Go is still enormous, so reducing the system model of Go is the key issue for improving the scale of verification. This is the motivation of this paper. Symmetry is a technique for reducing system model of model checking. Therefore, this paper analyzes the symmetry reduction in game verification.

Preliminary

Alternating-time Temporal Logic

In model checking, linear-time temporal logic (LTL) and branching-time temporal logic (CTL) can be used to specify requirements for reactive systems. However, for open systems whose behavior depends on the state of the system as well as the behavior of the environment, traditional temporal logic is insufficient. Alternating-time temporal logic (ATL) proposed by Alur enriches temporal logic
to specify alternating properties, which can represent requirements of open systems [4]. ATL formula can be used to specify properties which are always established no matter how external environment changes. For example, in a two player game, the property whether one can win the game no matter how the other responds, can be specified by an ATL formula.

In an ATL formula, X,G,U are traditional temporal operators, [], [ ] are path quantifiers. Suppose \( p \) is an atomic proposition, [\( \Delta \)]\( p \) specifies that all agents in A can cooperate to make \( p \) established, while [\( \forall \)]\( p \) means that even all agents in A cooperate cannot make \( p \) established.

**Symmetry**

Symmetry is one of effective techniques for alleviating state space explosion problem. Concurrent system often contains repeated parts. For instance, many protocol contains the network that consists of some isomorphic communicating processes; hardware devices also contain repeated components. If we can find out and combine the symmetrical states, simplified system will have equivalent properties with the original system, while the cost of verifying the simplified system will be effectively reduced.

**Game Model Checking Based on ATL**

In game formal verification, a game process can be modeled by an alternating Finite Automaton [7]. An Alternating Finite Automaton (AFA) is a nondeterministic finite automaton whose transitions are divided into existential and universal transitions. For an existential transition, automaton chooses to switch state to either state like a regular nondeterministic finite automaton; For a universal transition, all paths should be accepted like a parallel machine. Suppose \( \exists \) and \( \forall \) are two sides of an alternating game, \( S(\exists) \) is a finite state set of existential states and \( S(\forall) \) is a finite state set of universal states. each situation of game corresponds to a state in automaton, while each transition switches a state from one set to another. Beginning from a state in \( S(\exists) \), each accepted runs of automaton corresponds to a winning strategy of \( \exists \).

![Figure 1. The formal model of 2x2 go.](image)

Figure 1 gives an example for illustrating a process of playing 2x2 Go, where S with subscript indicates the states of automaton. In this automaton, \( S_0 \) corresponds to the empty board, which is the initial state of automaton; \( S_8 \) is a victory situation of Black, which is a terminal state. However, from the state \( S_0 \) of the automaton, Black can’t always enters victory situations without considering komi, thus \( S_0 \) isn’t a winning situation of Black. Since \( S_0 \) is the initial state of the automaton, Black don’t have a winning strategy in 2x2 Go.
ATL formula can be used to describe the winning strategy of a game. For example, ATL formula
\( \varphi = (\exists \mathcal{A}) \mathcal{X} p \) means that beginning from the current state of \( \exists \) side, no matter how \( \forall \) side responses, \( \exists \) side will reach the states satisfying \( p \) after a move; \( \varphi = (\exists \mathcal{A}) F p \) means that beginning from \( \exists \) side, no matter how \( \forall \) responses, \( \exists \) side will eventually reach the states satisfying \( p \) after a number of moves.

If a symbolic model checking algorithm determines that \( M \) satisfies \( f = (\exists \mathcal{A}) F p \), we can say that there exists a winning strategy for Black, where \( p \) means \( \exists \) wins, corresponding to all victory situations of A. All states satisfying \( f \) belong to all winning situations of Black, namely, beginning from these states, Black can always win. Based on fixed-point computation, symbolic model checking becomes more effective.

System Reduction Based On Symmetry

In the game tree, situations usually have a high degree of symmetry, while symmetrical situations have similar properties in general. With this support, if we can recognize and combine symmetrical situations in the game, the scale of system model will be greatly reduced.

In symbolic model checking, system states and transitions are represented as OBDD. In the process of transiting system model into OBDD, each state that we have processed will be recorded. When a new transition moves current state to another, if the state after the transition is same or symmetrical similar with a processed state, then we delete the new state and change the transition to the existed state. In this way, all the states of the system after reduction are no longer symmetrical similarity, thus the number of states is significantly reduced. The principle of the algorithm is shown in Figure 2.

```plaintext
unreached = \emptyset;
unexplored = \emptyset;
transitions = \emptyset;
for all initial state s do
    append \( \xi(s) \) to reached;
    append \( \xi(s) \) to unexplored;
done
while unexplored \( \neq \emptyset \) do
    remove a state s from unexplored;
    for all successor states q of s do
        if \( (s, \xi(q)) \) is not in transitions
            append \( (s, \xi(q)) \) to transitions;
        end if
        if \( \xi(q) \) is not in reached
            append \( \xi(q) \) to reached;
            append \( \xi(q) \) to unexplored;
        end if
    done
done
```

Figure 2. Symmetrical reduction algorithm.

From the algorithm we can see that, a method for recognizing symmetry states is critical. Fortunately, zobrist hashing provides such a way to determine whether two states are symmetrical. For a state in game process, we can transform the state variables according to the symmetry so that once we recognize a new state, all the symmetrical state are explored.
We take the game of Go as an example. The game of Go has a high degree of symmetry, each board situation in the fig has the same properties with others. According to this, we can apply the algorithm to reduce the system of Go. For the automation in Figure 3, after applying the algorithm, the reduced automaton are shown below. We can see the number of states are much less than before.

**Experiments**

We take CUDD package [8] as the representation of OBDD to implement the ATL symbolic model checking. We choose two traditional game -- tic-tac-toe and game of Go in small board as the examples for model checking to illustrate the validity of the reduction method.

**Tic-Tac-Toe**

For the two player game tic-tac-toe, each position has three states, thus each board situation can be represented by 19 boolean variables \((x_1, x_2, ..., x_9, y_1, y_2, ..., y_9, \text{turn})\). \(x_i=1\) indicates that player one occupied the \(i\)-th position while \(y_i=1\) indicates the other one; \(x_i=0\) and \(y_i=0\) indicate one occupied the \(i\)-th position; variable turn indicates the player who turns to move at current state.

Beginning from a empty board, all the variables are set to 0; Two players take turns occupying a empty position, each move transits the system from a state to another, until the board is full or a player has placed three of stones in a horizon, vertical or diagonal. We can see that the system can be represented by a automaton and the winning strategy for a player can be formalized by the ATL formula described earlier. Thus symbolic model checking can be used to verify the winning strategy of tic-tac-toe.

\[
\begin{align*}
000000001100000000 & \quad \text{1} \\
000000001010000000 & \quad \text{1} \\
000000001001000000 & \quad \text{1} \\
000000001000100000 & \quad \text{1} \\
000000001000010000 & \quad \text{1} \\
000000001000001000 & \quad \text{1} \\
000000001000000100 & \quad \text{1} \\
000000001000000010 & \quad \text{1} \\
\end{align*}
\]

\[
\text{And(InitState, All the win states of player1)} : \text{is the zero DD}
\]

Figure 4 shows the result of verification. From the figure we can see that the initial state is not in the set of player one’s winning situations, namely, player one don’t have a winning strategy from the beginning. However, if player one first occupied a corner position at first and then the opponent did not occupy the central position, then player one has a winning strategy.
Figure 5. Model checking result of Tic-Tac-Toe.

Figure 5 shows the comparison of the number of OBDD nodes before and after the reduction. From the result, we can see that the system node is significantly reduced, which illustrate the validity of the reduction algorithm.

**Go in Small Board**

Without considering komi, we implement symbolic model checking algorithm in 2x2 and 3x3 board, the results are shown in Figure 6. From the result we can see that in 2x2 board Black doesn’t has a winning strategy while in 3x3 board Black has a winning strategy.

![Image](image.png)

(a) 2x2 board (b) 3x3 board

Figure 6. Model checking result of go in small board.

With the help of the reduction method, the maximum size that our algorithm can search in ordinary PC increases to 5x5.

**Summary**

This paper mainly studies the application of symmetry technique in game model checking. Based on alternating finite automaton and ATL, symbolic model checking can be used to verify whether a player has a winning strategy under a game situation. In this way, we can use symmetry technique to reduce system model to improve the efficiency of model checking.

As a supplement of the MCTS Algorithms, deep learning, and big data, the model checking technique can avoid the lack of deterministic search of state-of-the-art algorithms to a certain extent. In addition, If the board can be divided into the stable parts and the unstable parts, thus we can implement model checking in at the unstable part. In such cases, the choices of the player is based on deterministic behavior instead of probability computation, which can maximize the benefit in the local range. This is the benefit of using the new approach.

**Acknowledgement**

This work has been supported by the National Natural Science Foundation of China under Grant No.U1204608, No.61250007 and No.61572444, as well as China Postdoctoral Science Foundation under Grant No.2012M511588 and No.2015M572120.
References

[1] D. Silver, A. Huang, C. Maddison, et al, Mastering the game of Go with deep neural networks and


672-713, 2002.

IEEE Transactions on computer Aided Design of Integrated Circuits and Systems, 13(4): 401-424,
1993.


-599.