Formal language and reasoning for playing Go

Arturo Yee Rendón and Matías Alvarado

Department of Computing, Center of Research and Advanced Studies, Mexico D.F.
ayee@computacion.cs.cinvestav.mx matias@cs.cinvestav.mx

Abstract. In this paper, we introduce a formal grammar for playing Go that fundaments an automated Go-player. Go-tactics such as eyes, ladders, mutual life and nets are properly modeled and tested. As well as Go-strategies to offensive or defensive purpose are introduced and their performance is tested throughout Go matches against humans or previous well-known automated Go-players like GNUGo. Results and a comparison analysis are reported and discussed on the perspective of current state of the art on Go automation.

Keywords: Game Theory, Go finite state machine, strategic reasoning.

1 Introduction

Originated more than 2,000 years ago, Go is a two-person perfect information game, and is one of the most complex board games. Writing computer programs to play Go is one of the grand challenges of Computational Intelligence nowadays. Each player's goal of the game is to control a larger area than the opponent's one on the board; the challenge for Go game automation is due to the simplicity of the pieces and rules to play it, hence the way to achieve the most board area control is a very open procedure, and the combinations for doing it have an exponential growth. Actually, the Go's search space of solutions is huger –very much– than the one of Chess [2].

The Go game is played on a board shaped with 19 horizontal and vertical lines commonly, see Figure 1, where alternating, each player places a stone of his own color on an empty intersection on the board, with black playing first [1]; black player takes the white stones and conversely. By following the each player's goal to control a larger board area than the opponent, one of the difficulties for both human and computer Go players is to determine when a group of adversarial stones is possible to capture, even the opponent makes any movement to save them. This situation is called unconditional life, and its determination is crucial for intelligent play 1.

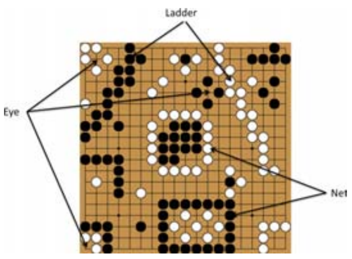


Fig. 1. Go board

In [3] the authors focus on evaluating non-final positions. In particular they deal with the task of estimating potential territory in non-final positions, which is much more difficult than determining territory in final positions. Prospective methods of programming the Go game will probably are of interest in other domains as well [5], which goal is to present the links between existing studies on Computer Go and AI related domains: evaluation function, heuristic search, machine learning, automatic knowledge generation, mathematical morphology and cognitive science.

Bounzy [8] build a cognitive model of human Go player called INDIGO, which aims to play full Go games as well as possible. It reads low level concepts: ladders, eyes and nets, see Figure 1. INDIGO recognizes territories as well as human players do, with mathematical morphology tools.

GNUGo is a free Go program that is able to play the Go games from board sizes of 5x5 to 19x19 [6]. GNUGo is deterministic. In our study, we developed a communication interface to test GNUGo with our simulator. The Smart Game File (SGF) was proposed in [6]. This format is simple tree which it represents the plays of player. SGF was originally designed for the exchange of Go records between humans and not for spreading computer oriented expert knowledge. Our simulator saves a game history of plays in SGF.

Description of the paper, it begins with a brief introduction to the formal rules of Go, in Section 2; we make a description of Go game simulator, in Section 3; the experimental stage, in Section 4 and conclusion; Ending with a discussion about future work.

2 Go game Description

A Go game can be represented as a tree, with each node corresponding to a particular board position. The root node is the position at the beginning of the game. The children of each node are the positions reachable in one move. A strategy for a player indicates how the player responds to any board position. A complete strategy for Go, even on the 9x9 board, would be astronomically large. The disarming simplicity of the rules of Go, however, conceals a formidable combinatorial complexity [7]. On a 19 x 19 board, there is approximately $3^{19 \text{ x } 19} = 10^{172.24}$ possible board configuration and, on average, on the order of 200 - 300 possible moves at each step of the game, compared with chess, chess has a much smaller branching factor, on the order of 35 – 40.

The player with the black stones is allowed to play first. It is only allowed to play one stone at the time, which is placed on one of the empty intersections. A stone, once played, is not to be moved unless if it is captured. If a stone or multiple stones of the same color are surrounded by the other color, such that no direct adjacent intersection is empty, and then the stone(s) is (are) captured. Adjacent empty intersections are also referred to as liberties. A liberty is an empty point adjacent to a group of stone. Any group that has no liberties is said to be dead and they have to be removed from board. From the capturing rule for multiple stones it can be seen that stones are connected which are positioned on directly neighboring intersections. Diagonal neighboring intersections are not connected as there is no direct line between them. Stones which are connected are also called a chain –a single stone is also a chain. The goal of the game is to control as much territory as possible. The player who has the largest territory at the end of the game is the winner.

The **tactics** in Go game are local conditions to deal with immediate fighting between stones [9], some tactics are describing next:

- An Atari is simple move to reduce the number of liberties.
- A ladder is a sequence of Atari to force the opponent into zigzag pattern and eventually the stones of enemy could be captured.
- A net is sequence moves that loosely surround some stones, preventing their escape in all directions, taking their liberties of enemy stones directly capture more easily on successive plays.

The **strategies** deal with global influence [9], calculating on the overall composition of the board and division of territory, and taking into account the influence of the stones on each intersection and given a tactical priority. Some strategies are listing.

- Mutual life happens when no player can play to a particular point without allowing the other player to play at another point to capture.
- Death is when stone(s) lacks living shape, meaning less than two eyes, and will eventually be removed from the board as captured.
- Invasion occurs when put a new living group inside an area where the opponent has greater influence.
- Reduction occurs when a player put a stone far enough into the opponent's area of influence to reduce the amount of territory.

3 Go Formal Modeling

In the next diagram and table, we show how the simulator works. In Figure 2 describes the process flow in the algorithm and in Table 1 the algorithm steps.

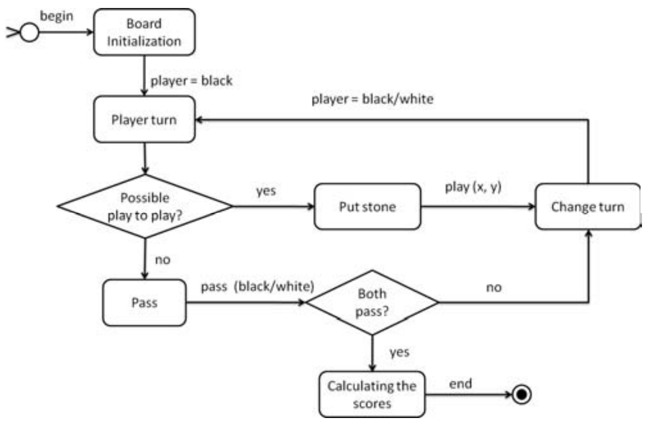


Fig. 2. General diagram

Table 1. Steps of Go game

Steps of Go game
1. Initializes the board
2. Player makes a move
3. If the move is possible go to step 5, otherwise go to step 4
4. Pass go to step 7
5. Put stone
6. Change turn go to step 2
7. If both players pass go to step 8
8. Calculate the scores and end game

Go Formal grammar. For the Go game, we proposed a simple formal grammar to help us to develop a Go game simulator and it is the following:

- V is the alphabet (terminals and non-terminals symbols).
- $\sum \subseteq V$ is the set of terminals.
- $B \in V - \sum$ *is the initial symbol.*
- $V - \sum$*, the set of non-terminal elements.*
- $R \subseteq (V - \sum) \times V^*$ *is the set of rules.*

Terminal symbols:

$\sum = \{play\ (player, x, y, t)\}$, where

- *player* ϵ {black, white}, *x* ϵ {0… 18}, *y* ϵ {0…18}
- *t* is the tactics/strategies to apply.
- *end* is the end of game.

Non-terminal symbols:

- *G* is the initial symbol.

- *J* is the play.
- *T* is the position played.
- *Pass-black* is when a player black pass.
- *Pass-white* is when a player white pass.

The set of rules:
- *G* → *J*.
- *J* → *T* | *Pass-black Pass-white end* | *Pass-white Pass-black end* | *Pass-black T* | *Pass-white T*.
- *T* → *play* (*player*, *x*, *y*, *t*).

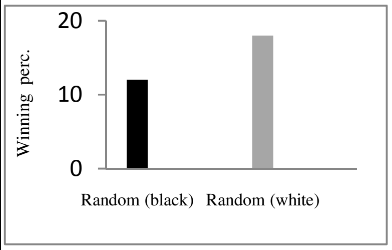
The Go game simulator is in Java and uses a graphic interfaces human – computer. Simulator is a heuristic method based on a formal grammar having a degree of variability by execution, sometimes giving an optimal result.

4 Experimental Results

In this section, we present some experiments of simulator, when each player plays with others; the experiments are the following: 1) random player (RP) vs. RP, 2) RP vs. smart player (SP), 3) RP vs. GnuGo player (GP), 4) SP vs. SP and 5) SP vs. GP. Nowadays thirty tests per experiment were made hence certain confidence was got. Additional more exhaustive tests will be made.

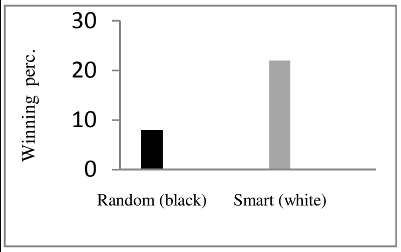
This paper has explored the diversity of tactics and strategies in the game of Go. We developed a simulator which is capable of playing competitive with the most traditional and powerful simulator of Go. From results is showed that when RP played with RP, both won at the same percentage, see Table 2; when RP played with SP, there was a clearly advantage to favor of SP, see Table 3; when RP played with GP, the first won more times than GP, see Table 4; when SP played with SP, both won at the same percentage, see Table 5; when SP played with GP, the SP won more times than GP did, see Table 6. We tested our simulator with GP and obtained a huge advantage to our simulator.

Table 2. Result of thirty runs RP vs. RP



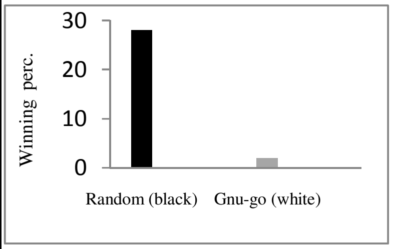
	Random player (black)	Random player (white)
1	55	44
2	46	74
3	48	55
. . .		
28	44	68
29	38	50
30	47	73

Table 3. Result of thirty runs RP vs. SP



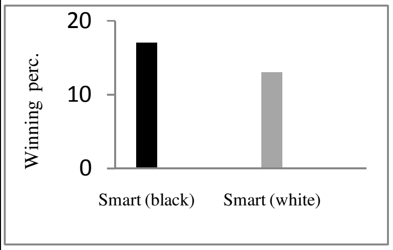
	Random player (black)	Smart player (white)
1	31	78
2	50	52
3	35	56
…		
28	36	67
29	61	50
30	39	67

Table 4. Result of thirty runs RP vs. GP



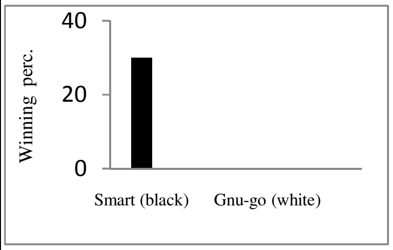
	Random player (black)	GnuGo player (white)
1	92	10
2	88	12
3	79	15
…		
28	96	10
29	89	12
30	57	28

Table 5. Result of thirty runs SP vs. SP.



	Smart player (black)	Smart player (white)
1	46	67
2	32	33
3	67	45
…		
28	71	43
29	46	80
30	79	39

Table 6. Result of thirty runs SP vs. GP.



	Smart player (black)	GnuGo player (white)
1	104	6
2	91	10
3	102	6
…		
28	99	6
29	102	6
30	99	66

***Conclusion*:** A formal grammar for modeling the Go game is introduced; the flow diagram and context-free grammar fundament the automated Go player algorithms. The deployed Go player mostly beat the well-known GnuGo, as well as to humans Go medium level of expertise or other automated Go players. The reason of the advantage is due to the application of offensive, territorial and defensive strategies introduced. The algorithmic implementation of these strategies, it supports the agile response by our automated Go player during the matches. Further test are required to assess the automated Go player performance.

***Ongoing work*:** The introduction of artificial neural network (NN) for patterns recognition allow acquire more information about the state of board game and what the enemy is doing. By segmenting the board game into 3 x 3 and 5 x 5 windows allows detect eyes, ladders, and net patterns. NN usage improves offensive/defensive tactics and strategies application for playing Go game. Moreover, for learning on the usage of tactics and strategies, given specific game circumstances. Formal grammar models the moves of the player and whole Go game, but still lacking for represent specific tactics and strategies, what we are working on.

Reference

1. D. B. Benson.: Life in the game of Go.: J. Information Sciences , 10(2),17-29, (1976).
2. K. Chen and Z. Chen.: Static analysis of life and death in the game of Go. J. Information Sciences, 121(1-2), 133-134, (1999).
3. M. Muller.: Computer Go. J. Artificial Intelligence, 134(1-2), 145-179, (2002).
4. E.C.D van der Werf, H.J. van den Herik and Uiterwijk.: Learning to estimate potential territory in the game of Go. Lecture Note in Computer Science, 3846, 81-96, (2006).
5. B. Bouzy and T. Cazenave.: Computer go: An AI oriented survey. J. Artificial Intelligence, 132(1), 39-103, (2001).
6. GnuGo. http://www.gnu.org/software/gnugo/gnugo_toc.html
7. E. Berlekamp and D. Wolfe.: Mathematical Go-Chilling gets the last point. Wellesley, MA: A K Peters, (1994).

8. B. Bouzy.: The INDIGO program. Proc. Of the 2nd Programming Workshop in Japan, 192-200, (1995).
9. Y. Nagahara.: Strategic Concepts of Go, Ishi Press, (1972).