The Game of Go: 
Bounded Rationality and Artificial Intelligence

Cassey Lee

ISEAS – Yusof Ishak Institute

cassey_lee@iseas.edu.sg

May 2019

Abstract

The goal of this essay is to examine the nature and relationship between bounded rationality and artificial intelligence (AI) in the context of recent developments in the application of AI to two-player zero sum games with perfect information such as Go. This is undertaken by examining the evolution of AI programs for playing Go. Given that bounded rationality is inextricably linked to the nature of the problem to be solved, the complexity of Go is examined using cellular automata (CA).

Keywords: Bounded Rationality, Artificial Intelligence, Go

JEL Classification: B41, C63, C70
The Game of Go: 
Bounded Rationality and Artificial Intelligence
Cassey Lee

“The rules of Go are so elegant, organic, and rigorously logical that if intelligent life forms exist elsewhere in the universe, they almost certainly play Go.”
- Edward Lasker

“We have never actually seen a real alien, of course. But what might they look like? My research suggests that the most intelligent aliens will actually be some form of artificial intelligence (or AI).”
- Susan Schneider

1. Introduction

The concepts of bounded rationality and artificial intelligence (AI) do not often appear simultaneously in research articles. Economists discussing bounded rationality rarely mention AI. This is reciprocated by computer scientists writing on AI. This is surprising because Herbert Simon was a pioneer in both bounded rationality and AI. For his contributions to these areas, he is the only person who has received the highest accolades in both economics (the Nobel Prize in 1978) and computer science (the Turing Award in 1975). Simon undertook research on both bounded rationality and AI simultaneously in the 1950s and these interests persisted throughout his research career. In 1992, Simon teamed up with Jonathan Schaeffer to publish an article titled “The Game of Chess” which surveyed how bounded rational humans...
play Chess and how computers are programmed to play the game. In the former case, Chess grandmasters were found to be able to remember and recognize Chess patterns (chunks) that lead to sound moves. The design of Chess computers can vary in terms of the combination of search and knowledge. Five years after the article was published, in 1997, the IMB’s Deep Blue supercomputer defeated the then reigning World Champion Garry Kasparov in a six-game Chess match.

When this benchmark was achieved in Chess, the game of Go was considered to be the last remaining challenge in the man-versus-machine competition in AI research. The game of Go was long considered to be a more complex game compared to Chess on account of the number of possible moves from the start of the game. In his survey on games computers, Schaeffer (2000, p.260) opined “it will take many decades of research and development before world-championship-calibre Go programs exist”. This feat was finally accomplished in 2016, when Google DeepMind’s AlphaGo defeated the Go world champion Lee Sedol 4-1. This watershed event clearly calls for a re-evaluation of bounded rationality and AI.

The goal of this essay is to examine the nature and relationship between bounded rationality and AI in the context of recent developments in the application of AI to two-player zero sum games with perfect information such as Go. This is undertaken by examining the evolution of AI programs for playing Go. Given that bounded rationality is inextricably linked to the nature of the problem to be solved, the complexity of the state space of Go is examined via a cellular automata (CA) perspective.

The outline of the rest of this essay is as follows. Section 2 will discuss how bounded rationality is related to AI. Section 3 examines the links between bounded rationality and AI from the perspective of the evolution of AI research on two-player zero sum games with perfect information. Section 4 examines the complexity of the game of Go from the perspective of cellular automata. Section 5 concludes.

3 The Game of Go is also known as Igo in Japan, Weiqi in China and Baduk in Korea. It has also been described as the “Surrounding Game” or “Encircling Game”. For a brief description of the game of Go, see Appendix 1.

4 In terms of topics not covered in this paper but might be worth pursuing in the future, several directions of enquiry are worth pursuing. One relates to combinatorial game theory (Albert et al. 2007). The other to the deep links between games and numbers (Conway, 2000).
2. The Intertwined Domains of Bounded Rationality and Artificial Intelligence

2.1 Bounded Rationality

What is rationality? And in what sense is rationality “bounded”? An agent is rational when he or she makes a choice to achieve an objective. In the mainstream economic theory of choice, rationality requires that such a choice be made in a consistent manner. Consistency here refers to the agent having preferences that conform to a set of axioms on completeness, reflexivity and continuity.

Bounded rationality is a departure from the above characterization of rationality as consistency. This point is clearly articulated in the 1950s by Herbert Simon who is often acknowledged to be the founder of bounded rationality. In his paper, Simon (1955) argued that humans in the real world experience a number of constraints that would limit the applicability of “global rationality” (rationality as consistency). These constraints can be classified into internal constraints (limited computational facilities) and external constraints (limited information about environment). Under such constraints, humans act in a bounded (approximate) rational manner by choosing actions based on heuristics that meet certain aspiration levels. They also learn by gathering information about the environment over time to improve their decision-making processes.

Another approach to bounded rationality focuses on the departure of real-world behaviour from “rationality as consistency”. Since the 1970s, psychologists and behavioural economists such as Kahneman, Tversky and Thaler have employed experiments to discover how people’s beliefs and choices differ from the optimal beliefs and choices in rational-agent models (Kahneman, 2003 and Thaler, 2018). Such departures from rationality as consistency characterize how rationality is bounded in reality. Though behavioural economists often cite Herbert Simon as an early pioneer in bounded rationality, their methodology differs from that of Simon in several ways. Behavioural economists employ primarily experimental methods which focus on outcome (action/behaviour). Simon does not rely much on experimental methods. Instead, he presents general observations on how people actually behave (e.g. play Chess). Simon is more interested in the cognitive limitations of humans, which also leads to artificial intelligence via the formulation of computational models/machines that simulate bounded-rational human decision-making.

5 Heuristics are discussed by Herbert Simon as well as Kahneman and Tversky.
2.2 Artificial Intelligence

The 1950s was not only significant for the formulation of bounded rationality by Herbert Simon. This period also saw the emergence of artificial intelligence (AI) as a full-fledged field of research (Nilsson, 2010). Simon was an active participant in some of the major AI events during this time, such as the Dartmouth Conference in 1956. AI, like today, was fairly heterogeneous methodologically then. This can be seen in the four categories of definitions of AI proposed by Russell and Norvig (2010), which continue to be valid descriptions to this day (Table 1).

Table 1: Four Categories of AI Definitions

<table>
<thead>
<tr>
<th></th>
<th>Human Behaviour</th>
<th>Rational Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Thinking</strong> (Mental Process)</td>
<td>1. Thinking Humanly</td>
<td>3. Thinking Rationally</td>
</tr>
<tr>
<td></td>
<td>Machines that think intelligently like humans</td>
<td>Machines that think rationally</td>
</tr>
<tr>
<td><strong>Acting</strong> (Action)</td>
<td>2. Acting Humanly</td>
<td>4. Acting Rationally</td>
</tr>
<tr>
<td></td>
<td>Machines that perform activities that human consider intelligent</td>
<td>Machines that act rationally</td>
</tr>
</tbody>
</table>

Source: Adapted from Russell and Norvig (2010), Figure 1.1, p.2.

In the first category of the definition, AI aims to construct computational machines (models) that possess thinking mechanisms (processes) that are similar to that of humans. These mechanisms include language, knowledge representation (and memory), reasoning and learning. In the second category, machines are merely required to act like humans. They do not need to possess human-like mechanism. In the third category, the focus is on machines that think rationally in terms of being based on mathematical logic. Finally, the fourth category relates to machines that take actions that are optimal (rational) but may not be based on logical reasoning. There is an alternative computational architecture of AI that is not based on mathematical logic but may lead to rational actions.

---

6 In this essay, the term machines include hardware and software (programs).
Hence, computational architecture is another area where there is diversity in AI methodology. In this respect, there are two major approaches to AI, each with its own lineage. Both these approaches were already present during the early years of AI in the 1950s. The first approach is based on mathematical logic while the second one is based on neural networks. The idea underlying the first approach, also known as von Neumann architecture, is based on the idea that intelligent action can be modelled using physical symbol systems. The second approach, neural networks, is based on building a computational architecture that is based on the human brain which comprises networks of connected neurons. Learning takes place in neural networks through changes in the strength of neural connections. Though the two traditions or approaches in AI have long been regarded as distinct, there are now attempts to bridge them (Kaiser et al, 2017).

2.3: AI: Towards Bounded Rationality or Rationality?

The literature on bounded rationality in economics does not usually make any references to AI. Similarly, AI scientists seldom discuss the concept of bounded rationality. Despite this, bounded rationality and AI are clearly intertwined. Drawing on the earlier discussions on the different types (definitions) of AI research, the approaches that attempt to model how humans think (Definition 1) and act (Definition 2) have to incorporate bounded rationality (Table 2). To be human is to be bounded rational - both human thinking and actions are bounded rational.

On the other hand, the other two approaches that focus on rationality (Definitions 3 and 4) are not bounded rational. Computational machines that incorporate processes that are global rational are a theoretical possibility. This possibility can be framed in terms of the Universal Turing Machine (UTM) – a hypothetical machine that uses a built-in rule table as a guide to reads and change an infinite strip tape written with symbols (e.g. 1,0). Such a machine can compute anything in the world. However, the UTM also demonstrates that it is not possible to tell whether, when and how a UTM will stop computing i.e. arrive at a solution. This problem – known as the Non-Halting Problem – demonstrates that although thinking rationally is theoretically attractive, acting rationally is incomputable. In other words, one can design a computer that will search for global optimality but it will not be possible to know whether global optimality will be attained.
From the above perspectives, the domain of AI is clearly oriented towards bounded rationality. One way to analyse this issue further is to examine this in terms of the application of AI to board games.7 This is examined next.


The links between bounded rationality and AI are most evident in AI research on two-player zero sum games with perfect information such as Checkers, Chess and Go. These links are evident during the early years of the founding of game theory and AI. The development of the Min-Max Theorem by John von Neumann in the late 1920s can be traced back to the then prevailing interests in set theory and Chess within the German-speaking mathematical community during this period (Leonard, 1995).8 The contributions of von Neumann straddle both economics (game theory and growth theory) and computer science (von Neumann architecture). Alan Turing (1953) – another founding father of computer science – dwelt on how computers can be used to play Chess. Herbert Simon simultaneously researched on bounded rationality and Chess-playing programs in the 1950s.

---

7 Griffiths et al (2015) provides an alternative discussion on how to relate models from algorithmic level to computational level using resource-rational analysis.

8 These early analyses of Chess using set theory is part of the axiomatization movement ala Hilbert which also influenced the mathematization of economic theories in the 1940s and 1950s.
As AI developed in subsequent decades after the 1950s, computer scientists continued to use two-player zero sum games with perfect information such as Checkers, Chess and Go as test beds for the application of AI. Progress in the application of AI in this area was often measured in terms of the ability of computer programs to beat world champions. An even more ambitious benchmark was to solve the game completely, which entails proving the result of a game played perfectly. Historically, the progress made in AI applications across the different board games has been uneven. This is not surprising as the various types of board games differ in their complexity. Go is often considered to be more complex than Chess while Chess is more complex than Checkers. But how should complexity be defined for such comparisons?

In the extant literature, complexity can be defined in two ways. First, space states complexity measures the number of positions that can be reached from the starting position (Bouzy and Cazenave, 2001, p.44). Second, game tree complexity measures the number of nodes in the smallest tree necessary to solve the game (ibid). Table 3 summarizes the complexity of various board games using these measures.

<table>
<thead>
<tr>
<th>Game</th>
<th>Number of positions reachable from stating position</th>
<th>Number of nodes in the smallest tree necessary to solve the game</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkers</td>
<td>10^{17}</td>
<td>10^{32}</td>
</tr>
<tr>
<td>Othello</td>
<td>10^{10}</td>
<td>10^{58}</td>
</tr>
<tr>
<td>Chess</td>
<td>10^{80}</td>
<td>10^{123}</td>
</tr>
<tr>
<td>Go</td>
<td>10^{160}</td>
<td>10^{400}</td>
</tr>
</tbody>
</table>

Source: Bouzy and Cazenave (2001), Table 1, p.44.

It is thus not surprising that Checkers was the first game for computers to beat world champions. This took place in 1990. By 2007, Checkers was completely solved with the discovery of draw as the final outcome of perfect play (Schaeffer et al., 2007). The more complex games of Chess and Go have not been solved until today. Though later than Checkers, computer programs have reached the level of being able to beat world champions – Chess in 1997 and Go in 2016.
Game-playing computer programs have clearly evolved over time with advances in computer software (algorithms) and hardware. Appendix Table 1 summarizes some of milestones in the application of AI to games. The progress made in AI-application to games have been possible in three key elements, namely: (i) knowledge; (ii) search; and (iii) learning. These elements are inter-related. Each of these will be discussed in the context of AI and bounded rationality.

3.1 Knowledge

Specific knowledge about a game is a key element of game-playing computer programs. For board games, this can take the form of historical games played (game database) and a set of expert-based strategies on how a game should be played for different states of the game (specific configurations of board during play). The game databases for Go usually comprise a few hundred thousand games played by professional players.\(^9\) This was made possible by advances in the coding of games in such a way that information in such games can be extracted by computers for analysis and learning. The extraction of information from games requires representation of knowledge. This, in turn, entails the encoding of Go knowledge. Muller (2002) suggests that there are two approaches to encode Go knowledge, namely, via patterns and structured knowledge representation.

In the first approach, the patterns on the board (Go positions) can be matched and compared with patterns in databases associated with good moves (which are identified by human experts). This matching process leads to the generation of moves in the game. The second approach involves a more abstract level of knowledge representation involving tactical and strategic concepts such as blocks (connected stones of same colour), type of connections between stones, and chains (blocks joined by pairwise connections).\(^10\) This approach also draws from human experiences of the game.

From a bounded rationality perspective, the representation of knowledge in Go – be it pattern or structural – explicitly involves human intelligence. It is a process of encoding the knowledge for machines as it is understood by humans. Experiments have shown that when playing board games, humans tend to remember chunks of patterns that are associated with

---

\(^9\) A list of such databases is available at: https://senseis.xmp.net/?GoDatabases

\(^10\) For more detailed explanations, see Muller (2002), pp.156-159.
potentially promising moves (Schaeffer and Simon, 1992). Given that there are many strategic dimensions and ways of winning a game, knowledge representation also requires an explicit specification of how they are related and whether there should be hierarchies of pattern valuations. These issues are also related to the notion of goals in the search of decision trees.

### 3.2 Search

The second element in AI-application to board games is search. Theoretically, two-player alternating games can be represented using minimax decision trees. In classical game theory, such games are solved by computing the minimax in a backward induction setting. However, it is difficult to use this method to solve for the game of Go due to the large size of the decision tree.\(^1\)

How large is the size of the decision tree for the game of Go? A 19x19 Go board has 361 grid intersections. After the player first moves, the second player can place his stone in one of the remaining 360 interactions, and so on. Beginner players are sometimes trained with 9x9 boards. The tree diagram for the first and second player using a 9x9 Go board is illustrated in **Figure 1** and **Figure 2**. Each node of the tree represent a state of the play. After Player 1 moves, Player 2 chooses one of the remaining 80 positions on the board. At this stage, there are 80 nodes connected to each of the 81 branches associated with Player 1. At this stage of the game, there are 6,480 branches (or 81 x 80). A crude estimate for the total number of branches for a 19x19 board is \(19! = 10^{768}\). The actual number of legal branches (moves) is smaller. The number of legal positions for a 19x19 Go board is huge. The exact size of it remains an open problem (Tromp and Farnebäck, 2007).

As it is computationally impossible to undertake a full brute-force search of the entire minimax decision tree for Go, the alternative is to search only selected trees (sub-trees). One selective search method that was discovered in the 1960s is the alpha-beta (\(\alpha\beta\)) algorithm. The alpha-beta algorithm essentially cuts off or prunes the minimax decision tree. The lower-bound or alpha refers to the minimum value that Player 1 has achieved whilst beta is the upper bound denoting the maximum value that Player 2 can limit Player 1 (Schaeffer, 2000). The minimax trees are pruned when alpha exceeds beta. However, the alpha-beta algorithm does have weaknesses – search is limited by fixed depth resulting in the size of the minimax tree to

\[\text{11 The minimax tree has two dimensions, namely, breadth (number of options at each stage of play) and depth (the number of game stages of play).}\]
grow exponentially with search depth (Kishimoto and Mueller, 2015). Furthermore, systemic errors can take place but remain hidden.

**Figure 1: Tree Diagram for Player 1’s First Move for 9x9 Go Board**

**Figure 2: Tree Diagram for Player 1 and 2’s First Moves for 9x9 Go Board**
Another approach to overcome the large size of the decision tree in Go is to employ statistical sampling of sub-trees in the games. The Monte Carlo (MC) method overcomes systemic errors by randomizing fixed policies that map states to actions. This is implemented by running a fixed number of simulations from an initial state (board configuration) until the end of the game (Gelly et al., 2012). Such simulations generate expected values for each of the different states.

An important advance in the application of AI to Go was the discovery of the Monte-Carlo Tree Search (MCTS) algorithm, which is a hybrid approach that combines Monte-Carlo simulations with tree search.\(^{12}\) The MCTS simulations achieve two things – it “grows” the decision tree and improves the accuracy of the expected values of the nodes. An effective modification of the MCTS algorithm involves the incorporation of an exploration bonus to the expected value in the evaluation (scoring) function. This encourages the exploration of least-tried actions.

It is clearly not possible to undertake a brute-force minimax tree search for global optimality in complex games such as Go. AI researchers have attempted to improve search algorithms by limiting search to a subset of trees that lead to approximate global optimality. This was done either by pruning the decision tree searched or by probabilistically sampling and growing of decision trees. Thus, even though the computer programs were designed to act (play) as (bounded rational) humans, computational limitations constraint the adoption of rational thinking processes (which would have involved minimax brute-force search). It is also important to note that the nature of search algorithms employed are of machine bounded rational and non-human bounded rational. Non-human bounded rationality can perform better than human champions (e.g. Deep Blue vs. Kasparov) or poorer than human champions (in Go competitions before 2016). The latter was only achieved with further advances in AI that incorporated learning mechanisms which represent (after knowledge and search) the third important element of AI-applications to games.

\(^{12}\) See Gelly et al. (2012, p.109) for a description of how the MCTS works.
3.3 Learning

Learning entails the acquisition, consolidation and retrieval of information. Though not unique to the human species, learning is one of its key characteristics. Go players undertake years of learning to become professional players. As discussed earlier, professional Go players can use their knowledge to influence the design of Go playing computer programs. Machine learning refers to something else – the learning process is undertaken by computers. Even though machine learning has become a key element in computer programs capable of beating world champion Go players, learning per se is not a new element in AI. Machine learning was already an active area of research in the 1950s but it did not take-off until the 1980s. A key reason for this was the dominance of the logic-based approach to AI and the von Neumann architecture. These trends affected how AI was applied to board games such as Chess and Go.

Machine learning can be broadly divided into two types – supervised learning and unsupervised learning. Machine learning is often implemented using neural networks – in which multiple computing cells or processing units (either in the form of hardware or software) are inter-connected. The strength of these interconnections, as measured by the value of the synaptic weights, changes during the learning process. An example of supervised learning would be programmers choosing the set of professional Go games to train neural networks. Knowledge representation is still needed in the form of specifying what is to be learned. Unsupervised learning is machine learning that involves automatic generation of knowledge. Go programs involving self-play falls into this category of machine learning. Early Go-playing programs in the 1990s used a variety of learning approaches – some employed supervised learning while other used unsupervised learning via stochastic self-play (Bouzy and Cazenave, 2001). Despite these development, Go-playing programs using machine learning achieved moderate success as these programs were not strong enough to challenge professional Go players, let alone a Go world champion.

It took another two decades before a Go-playing program could finally defeat a Go world champion. This took place in 2016 when Google DeepMind’s AlphaGo defeated the world champion Lee Sedol 4-1. In 2017, an even more startling achievement was announced when AlphaGo Zero - an AI program starting from zero knowledge (without human guidance)

---

13 Other advances such as temporal difference learning – a reinforcement learning algorithm - also helped improve Go-playing programs in the 1990s. See Schaeffer (2000).
14 See Silver et al. (2016) for technical details.
and based on reinforcement learning - defeated AlphaGo 100 games to zero.\textsuperscript{15} Subsequently, a more generalized version of the AlphaGo Zero program was developed (AlphaZero) that was capable of self-learning, and capable of defeating world champion programs in the games of Chess, Shogi and Go.

The success of AlphaGo is possible due to cumulative advances in AI such as Monte Carlo Tree Search (MTCS) and deep convoluted neural network (ConvNet). ConvNet is a deep-learning architecture involving the stacking of multiple layers of neural networks (LeCun, et al., 2015). This computational architecture enables multi levels of representation by extracting different features of an input (e.g. Go board configurations). AlphaGo is essentially a hybrid program that first implements supervised learning by training ConvNets on human expert moves in professional games. The outputs from this is then fed into unsupervised (reinforcement) learning that entails stochastic self-play (Silver et al., 2016). The program generates two key outputs: (i) valuations of subtrees (i.e. payoffs) - which can be used to reduce the depth of search, and (ii) policy networks – high probability moves that reduces the breadth of search through sampling of sub-trees.

The next advancement – AlphaGo Zero – is a pure self-play (reinforcement learning) and MCTS search program that does not rely on any human knowledge (Silver et al., 2017). Other characteristics of the program’s architecture include raw board representation as inputs, combination of policy and value networks, and tree search without growing trees via Monte Carlo rollouts. Based on the analyses of the games played by AlphaGo Zero, the program clearly exhibits Go-playing capabilities that surpass human players. Silver et al. (2017) noted that the program is able to re-discover basic as well as discover new (non-standard) Go knowledge.

What is interesting about AlphaGo and AlphaGo Zero is that in both programs “thinking” is bounded rational in that it is not designed to undertake a full search of the decision tree. These programs are, however, not human bounded rational but machine bounded rational. The super-human game play demonstrated by AlphaGo and AlphaGo Zero also demonstrates that machine bounded rationality can lead to super-human artificial intelligence, at least in Go playing. However, this is not to suggest that bounded rational machines are superior to bounded rational humans in all respects. The two are different. Human bounded rationality is the product of evolution and is physically (biologically) embodied (at least for now). Two

\textsuperscript{15} See Silver et al. (2017).
lines of enquiries are possible. The first, along the lines of Herbert Simon’s research program which emphasizes cognitive limits (computation, memory). Second, it would be interesting to investigate how human’s bounded rationality along the lines of classical behavioural economics affect human’s game play.

The game of Go is also not a suitable Turing-type test for AI. Simon (1956) has long emphasized the intertwining of bounded rationality and the environment (problem to be solved). The environment of the game of Go is finite, fixed and deterministic. It is possible to argue that computers are likely to have the upper hand in discovering universes with deterministic rules such as in Go. To further understand this, it is perhaps useful to explore the complexity of the discrete universe of the game of Go.

4. The Complexity of Go: A Cellular Automata Perspective

The board for the game of Go can be represented as a lattice of square cells similar to that of a two-dimensional cellular automata (CA) (Figure 3). The configuration or pattern of a Go board during play corresponds to the state of cells in CA. This leads to the question of whether it is possible to think about Go in terms of CA and if yes, what type of insights can be gained from doing so. This should begin with a description of what is a CA.

---

16 See Lee (2011) for further elaboration of this view.
The CA is a computational model made up of a lattice network of cells. Though the cells are depicted as squares in Figure 3, the cells of CAs can take other shapes such as triangle and hexagon. The significance of these cell shapes lies in the number of neighbours that a cell can be connected to and interact with. Computation is carried through the simultaneous change in the states of cells brought about by local interactions with neighbouring cells. Such interactions are specified by a local transition function. For example, in a one-dimensional CA, the state of cell \( i \) or \( c_i \) at time \( t+1 \) can be represented as:

\[
c_i(t+1) = c_{i-1}(t) + c_i(t) + c_{i+1}(t) \mod 2
\]

where \( c_{i-1} \) is the state of cell \( i \)'s left neighbour, \( c_{i+1} \) the state of its right neighbour and \( \mod 2 \) the remainder after division of the sum by 2.\(^{17}\) This can be graphically represented as:

\[
\begin{array}{ccc}
  & C_{i-1} & C_i & C_{i+1} \\
\end{array}
\]

The transition function for a cell \( c_{i,j} \) in a two-dimensional CA with interactions with four neighbours (the von Neumann model) can be expressed as:

\[
c_{i,j}(t+1) = c_{i-1,j}(t) + c_{i+1,j}(t) + c_{i,j}(t) + c_{i,j-1}(t) + c_{i,j+1}(t) \mod 2
\]

The graphical representation for this is depicted in below:

\[
\begin{array}{ccc}
  c_{i,j-1} & & \\
  c_{i-1,j} & c_{i,j} & c_{i+1,j} \\
  & c_{i,j+1} & \\
\end{array}
\]

The dynamical evolution of the states of CA cells is driven by the rules in the transition function. This has led to the classification of CA into four types:

- **Class I**: Fixed Point – where CA cells evolve to a uniformly constant state
- **Class II**: Periodic – where CA cells evolve toward continually repeating a periodic structure
- **Class III**: Chaotic – where CA cells evolve randomly
- **Class IV**: Complex – where CA cells evolve in such a way that localized structures are produced that move about and interact with each other

\(^{17}\) See Schiff (2008), p. 43.
Graphical depictions of the four classes for one-dimensional CAs are presented in Figure 4.

Even though researchers have identified four classes of CA, questions relating to long term evolution of CAs are undecidable (Kari, 2012). In other words, while we know there are four classes of CA, it is not possible to a priori and comprehensively sort out which rule (transition function) falls into each of the four classes. This is not surprising as the CA is a universal computing machine and hence is also subject to the Non-Halting Problem identified.
for Universal Turing Machines. The implications of this for the game of Go is, indeed, intriguing.

The play of Go, seen from the CA perspective, is akin to a computational process. Unlike CA, however, there appears to be no equivalent transition function that governs how a Go gameplay evolves. The closest to a transition function would be the rules of Go and the goal of territorial control. AI programs that engage in self-play explore the landscape of Go based on these rules and goal. The undecidability of CA evolution, however, do seem to suggest that even though AI programs may achieve super-human capabilities in Go, it is not be possible for these programs to solve the game of Go by way of self-play in such a way that they will identify a single self-play game that produces global optimality. Thus, AI machines are always going to be machine bounded rational and such machines, while able to defeat bounded rational human world champions, will intrinsically never be able to solve the game of Go.

5. Conclusion

The game of Go was, for many years, the benchmark for the contest between human intelligence and artificial intelligence (AI). The defeat of a Go world champion by Google DeepMind’s AlphaGo in 2016 has re-ignited the debate on whether machine intelligence have finally surpassed human intelligence. Given that human intelligence is bounded rational, how is AI related to bounded rationality? AI researchers have defined AI in terms of a framework that has two dimensions, namely: (i) thinking versus acting, and (ii) human versus rational behaviour. If human rationality is characterized as bounded rationality, there is then a class of AI that are bounded rational either in terms of processes (thinking) or results (acting).

The links between bounded rationality and AI are most evident in the applications of AI in two-player zero sum games with perfect information such as Checkers, Chess and Go. The three key elements in these applications are knowledge, search and learning. A closer examination of the evolution of Go-playing programs along these three elements suggests that these programs are machine bounded-rational in a way that is different from human bounded rationality. The advent of advanced self-playing Go programs that do not rely on human knowledge further prompt questions about bounded-rational machines’ ability to explore the game-play landscape of Go exhaustively. To think about this problem, it might be useful to frame Go in terms of cellular automata. In other words, a Go game play (including self-play)
can be thought of as a computational process. The undecidability in CAs seems to suggest that bounded-rational Go-playing machines, whilst able to surpass bounded-rational humans, may not be able to solve the game of Go.

**References**


Appendix 1: Rules of Go

Go is a two-player game played on a board with a 19 x 19 grid. Novice players often train with smaller-sized board with 9 x 9 grids or 13 x 13 grids. Stones of two colours, black and white, are placed alternatingly at the grid intersections. The game begins with the player with the black stone placing his/her stone on an empty board. A player can pass his/her turn at any point of the game but consecutive (back-to-back) passes ends the game. One or more stones of a given colour (say white) is “captured” when it is completely surrounded by stones of the other colour (black). Stones that have been placed on the board remain there until they are captured. Players are not allowed to “commit suicide” by placing their stones at intersections that immediately results in their stones being captured by default. Moves resulting in repetitive capture are also not allowed. The game ends when a player resigns or when both players pass their turns consecutively. Note that the person who makes the last move is not necessarily the winner. The winner of the game is the player who controls the most territory at the end of the game. The game can also end in a draw.
# Appendix Table 1: Some Key Developments in Game-Playing Computer Programs, Computer Science and Mathematics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to 1930s</td>
<td></td>
<td>• 1845 - Charles Babbage first discussed computer program to play Chess</td>
<td></td>
<td>• 1937 - Claude Shannon’s master thesis containing ideas on Boolean algebra and binary arithmetic</td>
<td>• 1902 - Charles Bouton published mathematical paper on Nim</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1913 - Ernst Zermelo proved that chess has pure optimal strategies</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1937- Claude Shannon’s master thesis containing ideas on Boolean algebra and binary arithmetic</td>
<td></td>
<td></td>
<td>• 1928 – John von Neumann proves the Minimax Theorem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1935 – Roland Sprague and Patrick Grundy independently published papers on impartial games</td>
</tr>
<tr>
<td>1940s</td>
<td>• 1940s - Konrad Zuse design Chess-playing programs</td>
<td></td>
<td></td>
<td>• 1943 – Warren McCulloch and Walter Pitts published paper on neural networks</td>
<td>• 1946 – Stanislaw Ulam discovers the Monte Carlo Method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1949 – Richard Guy solved Dawson Chess and re-discover Sprague-Grundy Theory</td>
</tr>
<tr>
<td>1950s</td>
<td>• 1945 – Christopher Strachey completed first checkers program</td>
<td>• 1951 – Christopher Strachey completed first checkers program</td>
<td>• 1951 – Alan Turing published papers on Turing machine and Turing Test</td>
<td>• 1950 – Alan Turing published papers on Turing machine and Turing Test</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1952 – Arthur Samuel completed his first operating Checkers program</td>
<td></td>
<td>• 1954 to 1955 – Alan Newell, J.C. Shaw and Herbert Simon develop programming language</td>
<td>• 1954 to 1955 – Alan Newell, J.C. Shaw and Herbert Simon develop programming language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1959 – Arthur Samuel’s paper on a computer program that can defeat strong human players</td>
<td>• 1953 – Alan Newell and Herbert Simon published paper on Logic Theory Machine</td>
<td>• 1955 – Oliver Selfridge published paper on pattern recognition</td>
<td>• 1955 – Oliver Selfridge published paper on pattern recognition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1950 – Claude Shannon’s paper on Chess-playing program</td>
<td></td>
<td></td>
<td>• 1956 – Allen Newell and Herbert Simon published paper on Logic Theory Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1953 – Alan Turing’s paper on a computer program for playing Chess</td>
<td></td>
<td></td>
<td>• 1957 - Frank Rosenblatt invented Perceptron</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1958 – Newell, Shaw and Simon’s survey paper on computer programs for Chess</td>
<td></td>
<td></td>
<td>• 1958 - Allen Newell and Herbert Simon published paper on General Problem-Solving Program</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1958 – John McCarthy implement programming language LISP</td>
<td>• 1958 – John McCarthy implement programming language LISP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1953 – John Milnor published the first theoretical paper on Partizan games</td>
</tr>
<tr>
<td>Year(s)</td>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1960s</td>
<td>• 1963 – Arthur Samuel’s machine learning program beat a master level player</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1960 – David Leikovitz wrote the first Go computer program</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1963 – H. Remus published the first paper on Computer Go</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1968 - Albert Lindsey Zobrist’s PhD thesis on pattern recognition and Go</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1960s – Alpha-Beta algorithm discovered</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1970s</td>
<td>• J. Ryder’s PhD thesis on heuristic analysis of trees in Go</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1975 – Donald Knuth and Ronald Moore’s first published paper on Alpha-Beta Pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1970 – John Conway invents Game of Life</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1976 – Publication of Numbers and games by John Conway</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1980s</td>
<td>• 1989 - Efforts to design a program to beat world champion began</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1987 - Ing Cup, the international Computer Go competition launched</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1982 - John Hopfield proposed the idea of a network with bidirectional lines</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1985 - T.L. Lai and Herbert Robbins – provided an analysis of optimal solution for multi-arm Bandit problem</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1987 – Bruce Abramson PhD These containing ideas on Monte Carlo Tree Search (MCTS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1988 – Richard Sutton publishes paper on temporal difference learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1990s</td>
<td>• 1992 - The Chinook program was narrowly defeated by Marion Tinsley – the best human checkers known</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1994 – Chinook drew with Marion Tinsley</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1997 – IBM’s Deep Blue defeats World Chess Champion Garry Kasporov</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1993 – Brugmann’s report on one- ply Monte Carlo Go</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1998 – Publication of paper by Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner applying convolution networks to document recognition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000s</td>
<td>• 2007 – Schaeffer et al solved Checkers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2006 – René Coulom paper on MCTS algorithm,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2015 – Google DeepMind’s AlphaGo defeated European Go champion Fan Hui 5-0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2016 - AlphaGo defeated the world champion Lee Sedol 4-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2017 - AlphaGo Zero defeated AlphaGo 100-0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• AlphaGo Zero can play Chess, shogi and go at World Champion-level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2002 – Peter Auer, Nicolò Cesa-Bianchi and Paul Fischer introduced the upper confidence bound (UCB) for exploration versus exploitation dilemma</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>