Combinatorial Games and Machines

Lisa Rougetet

Abstract Mathematics and technology interact with each other for their respective benefit. This article will try to illustrate this unavoidable fact by studying the machines built to play games—especially combinatorial games (no chance moves)—and their impact on the development of mathematical ideas. We will see how all began with the simple Nim game and the first machines built to play it (and to win!). Then we will focus on the game of Chess to demonstrate that some mathematical ideas can originate from technologies and that these technologies can enable their concrete achievement.

Keywords Mathematics · Combinatorial game theory · Nim game · Chess · Technological advances in programming · Artificial intelligence · Minimax algorithm · Alpha–Beta pruning · Backward induction · Chess automaton · Game analysis · Complexity · Selectivity · Data storage

1 Introduction

The emergence of artificial intelligence cannot be precisely dated. Indeed, this theme of reflexion is quite old and can be found regularly through history with myths about artificial creatures, construction of automatons, or first attempts to formalize human thoughts. The term "Artificial Intelligence" (AI), coined in 1956 by John McCarthy (1927–2011), can be defined as the construction of computer programs that try to solve tasks that are, at the moment, better fulfilled by humans as they require high-level mental processes. These programs can be created for reasoning, for the understanding of natural languages, for perception, or for example, for games and mathematical practice. We will now focus on these last two points, especially through combinatorial games and through the game of Chess which was the starting point of many reflections about human thinking processes.

L. Rougetet (✉)
Laboratoire d'Informatique Fondamentale de Lille, Lille 1 University Science and Technology, Villeneuve d'Ascq, France
e-mail: lisa.rougetet@gmail.com

© Springer Science+Business Media Dordrecht 2015

R. Pisano (ed.), *A Bridge between Conceptual Frameworks,* History of Mechanism and Machine Science 27, DOI 10.1007/978-94-017-9645-3_24

2 First of All: What is a Combinatorial Game?

In this article we are dealing with special games called combinatorial games. They are defined by properties which are the following (Berlekamp et al. 2001, p 14):

1. There are just two players, often called Left and Right.
2. There are several, usually finitely many, positions, and often a particular starting position.
3. There are clearly-defined rules that specify the moves that either player can make from a given position to its options.
4. Left and Right move alternately, in the game as a whole.
5. Both players know what is going on, *i.e.* there is complete information.
6. There are no chance moves such as rolling dice or shuffling cards.
7. In the normal play convention a player unable to move loses.
8. The rules are such that play will always come to an end because some player will be unable to move. This is called the ending position. So there are no games that are drawn by repetition of moves.

These properties give a strict definition of a combinatorial game. However a more tolerant conception of combinatorial game theory does not always respect the seventh and the eighth points (note for example, that Chess does not fulfil the seventh condition as a match can end up in a draw). The global aim of the combinatorial game theory is to study the nature of the several positions of the game (winning, losing or draw) in order to build a strategy that will lead to a win. Let us now focus on the Nim game, a combinatorial game that enabled great improvements in the mathematical theory of games and for which a machine was specially built to play it.

3 The Starting Point: The Nim Game

The Nim game is one of the most famous combinatorial games. It was introduced for the first time under this name[1] by a mathematician from Harvard, Charles Leonard Bouton (1901) (1869–1922) in an article of the famous journal *Annals of Mathematics*, published in 1901 (Bouton 1901). This article is considered as the starting point of the relatively recent (twentieth century) mathematical theory of combinatorial games. Indeed its mathematical content is at the base of the resolution of the more general class of impartial games[2].

[1] Nim comes from the imperative form of the German verb nehmen, which means to take.

[2] An impartial game is a combinatorial game in which the available moves are similar for both players (which is not the case in Chess for example). The main theorem about the resolution of impartial games was independently found by Roland Parcival Sprague (1894–1967) in 1935 and Patrick Mickael Grundy (1917–1959) in 1939, and states that "every impartial game is just a bogus Nim-heap" (Berlekamp et al. 2001, p 56). This means that, thanks to some modifications

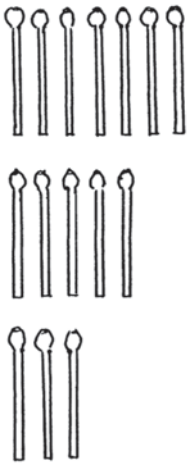

Fig. 1 A possible starting position at Nim. (Eiss 1988, p 188)

Because its resolution is based on the binary system, it was not very difficult to elaborate a program able to play it. This is what we will see just after presenting the rules of the Nim game.

3.1 Presentation of the Nim Game (Bouton 1901)

Bouton justifies his interest to the Nim game on account of its seeming complexity despite its extremely simple and complete mathematical theory. Then he describes the game as followed: upon a table opposing two players, A and B, three piles of objects of any kind are placed, let us say matches. The number of matches in each pile is arbitrary, except that it is well to agree that no two piles shall be equal at the beginning (see Fig. 1 for a possible starting position). Bouton requires this latter condition but actually it is not necessary to respect, for it does not change anything in the resolution of the game.

Alternatively, players select one of the piles, and take from it as many matches as they want: one, two,…, or the whole pile. The first one who takes the last match or matches from the table wins the game.

The entire theory of the Nim game is based on the notion of "safe combinations" (Bouton 1901, p 35). It refers to special positions that allow[3] the player who reaches one of them to win the game at the end (under the condition of playing "without mistake" (Bouton 1901, p 35)). Safe combinations present the following properties (Bouton 1901, p 36):

---

and transformations, we can always identify an impartial game to a determined Nim game (or a succession of Nim games).

[3] It is not necessary for the player who reaches a safe combination during the play to win the entire game. To do that, he must reach a safe combination at every move he makes.

> THEOREM I. If A leaves a safe combination on the table, B cannot leave a safe combination on the table at his next move. […]
> THEOREM II. If A leaves a safe combination on the table, and B diminishes one of the piles, A can always leave a safe combination.

These theorems and their proof provide the explanation of a possible win when a player reaches a safe combination.

Now, how can we check if a position is a safe combination or not? First, Bouton (1901, pp 35–36) explains that the numbers of matches of the piles have to be written in the binary scale. Then these binary numbers are placed in three horizontal lines so that the units are in the same vertical column. Then if the sum of each column is congruent to 0 mod. 2, the set of numbers on the table forms a safe combination. It is called the Nim-sum. In our example, 7, 5, 3 is not a safe combination and to change this, we would have to remove a single match from one of the three piles[4]. It is up to you now!

3.2 The Nimatron

In the Spring of 1940, an electromechanical Nim player machine (see Fig. 2) called *The Nimatron*, invented by two members of the staff of the Westinghouse Electric Company during their lunch break (Condon 1942, p 331), was built and exhibited at the Westinghouse Building of the New York World's fair, where it played more than 100,000 games (and won 90,000 of them). The *Nimatron* could play a game made up of four piles containing at most seven counters.

Condon underlines that the *Nimatron* is not like the other mathematical machines and that it serves no other useful purpose than entertainment, "unless it be to illustrate how a set of electrical relays can be made to make "a decision" in accordance with a fairly simple mathematical procedure". (Condon 1942, p 330) (Fig. 3).

3.3 Redheffer's Machine

Despite the fact that the *Nimatron* had no other aim than to entertain the visitors, the construction of a much more improved Nim-playing machine started in 1941. It was designed by Raymond Moos Redheffer (1948, p 343) (1921–2005), an assistant professor of mathematics at the University of California at Los Angeles (Gardner 1959, p 156) who stated "this theory is of such a nature that the computations required can be carried out by simple electrical circuits" (Redheffer 1948, p 343). Redheffer's machine proposed the same arrangement as the *Nimatron*, the equivalent of four piles containing seven counters at most, but weighed only 2.3 kg (5 pounds) against a ton for the *Nimatron*!

[4] The exercise is let to the reader!

Fig. 2 The *Nimatron* is a machine which is very skilful at playing the game of Nim (Condon 1942, p 330)

Redheffer extended his results in a B.S. Thesis in Mathematics to a more general game where players can remove objects from k piles and not only from one[5].

The first task of Redheffer's machine is to convert the number of objects of each pile in the binary system. Each of these numbers is represented by a switch supplied with as many layers of contacts ("pies") as there are digits in the binary number required for representing the maximum number of objects in the pile (see Fig. 4).

Then, the next step is to find the sum of the converted numbers. Redheffer simplifies the problem to an arrangement of connected switches (see Fig. 5): "It follows that E and G will be connected, and F and H will be connected, whenever an even number of switches are in the down position" (Redheffer 1948, p 347).

As the machine was planned for a maximum of seven objects in four piles, there are four switches, each having eight positions. Since any numbers between 0 and 7 can be written with three digits in the binary system, there are three indicator lights for each pile. "They are grouped behind a translucent screen covering a large round hole above each switch.

[5] This version of the Nim game is actually due to an American mathematician, Eliakim Hastings Moore (1862–1932), who extended in 1910 the work of Bouton and gave a generalisation of the Nim game when the player can remove matches from k piles. (Moore 1910).

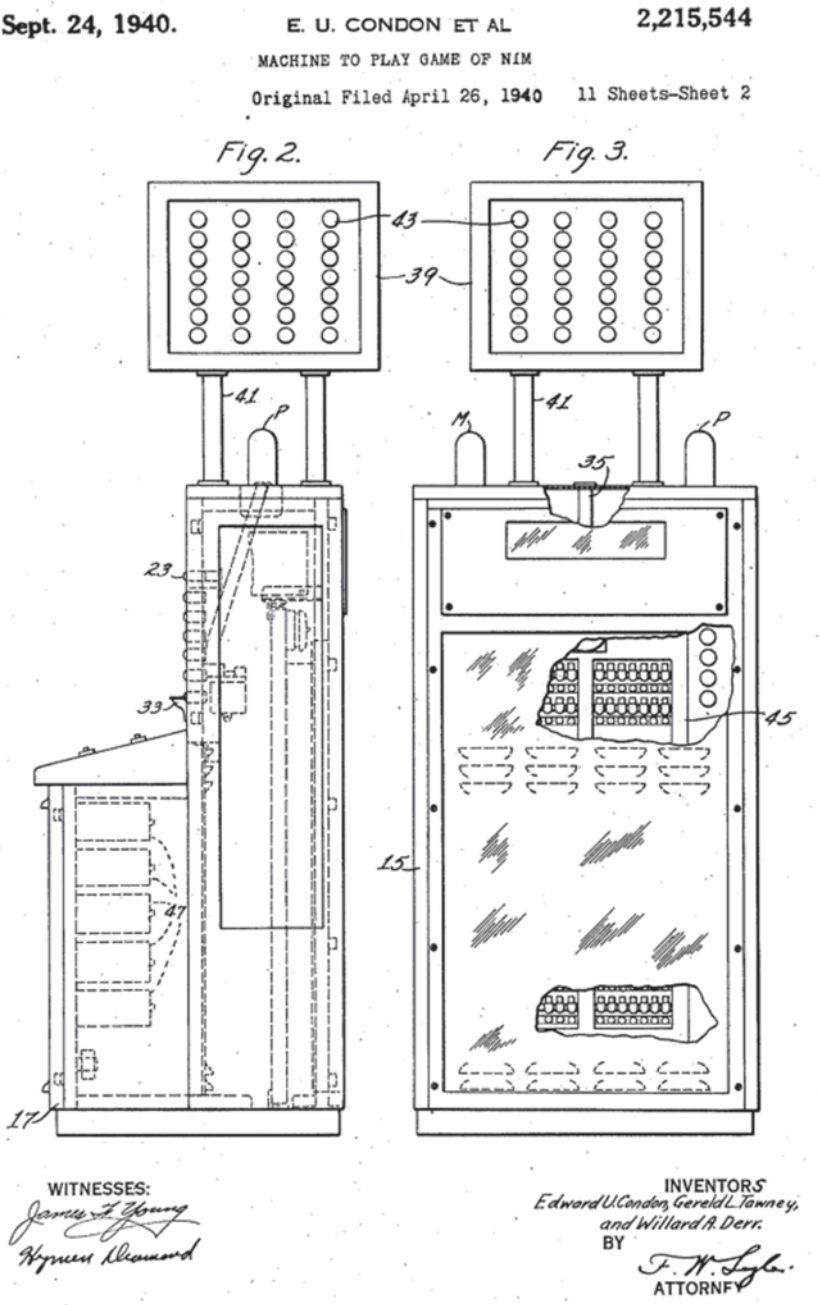

Fig. 3 Description of the patent of the *Nimatron* (US Patent 1940, p 2)

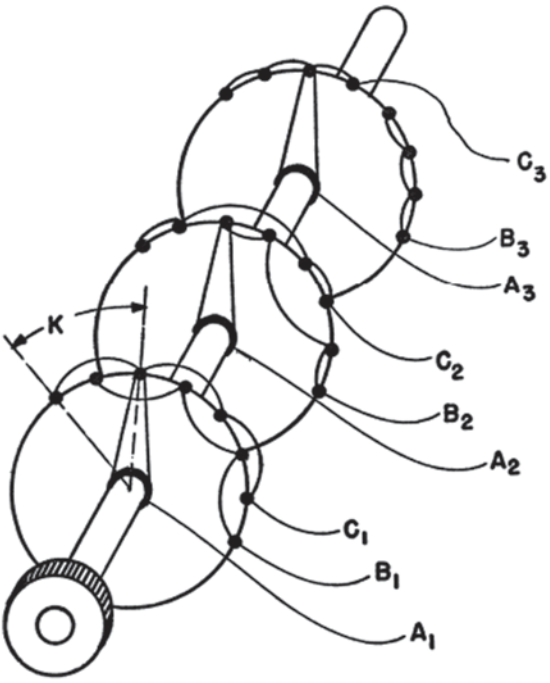

Fig. 4 This figure shows the connections for the pies representing the unit's, two's, and four's digits. (Redheffer 1948, p 345)

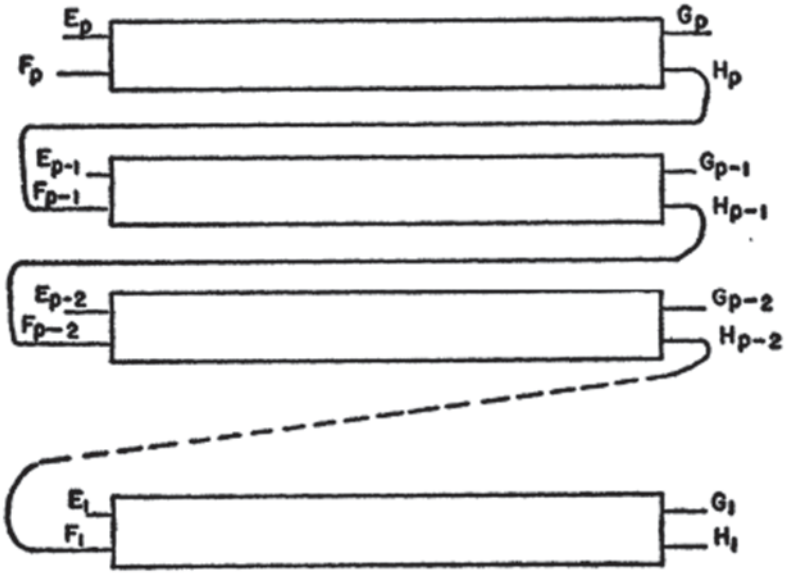

Fig. 5 "[…] but the connections will be interchanged whenever an odd number of switches are in the down position." (Redheffer 1948, p 347)

Neon bulbs are used partly because of their low current requirement and partly because they will not light if two are in series. The input is 110 V." (Redheffer 1948, pp. 348–349). As far as we know, Redheffer's machine was not exhibited in public, so no results about the games it played are available.

Redheffer's machine is a good example that shows the technological evolution (less or new material, lighter components) of a mechanical device initially designed around a mathematical idea.

3.4 The Nimrod

A few years later, Ferranti, the electrical engineering and defence electronics equipment firm, designed the first digital computer exclusively dedicated to play Nim, *The Nimrod*. It was exhibited at the Festival of Britain (Exhibition of Science) in May 1951 and afterwards at the Berlin Trade Fair (Industrial Show) in October. Its exhibitions were a great success and few witnesses relate that the most impressive thing about the *Nimrod* was not to play against the machine but to look at all the flashing lights which were supposed to reflect its thinking activity! (See Fig. 6) Some even say that none of the persons who came to play against *Nimrod* noticed the British bar next door, which offered free beverages… (Gardner 1959, p. 156). This particular display was built on purpose to illustrate the algorithm and the programming principles involved.

After the *Nimrod*, it seems that no other machines were built to play Nim. The simplicity of the game and of its solution may have led the scientists to turn to more complicated games both in their rules and in their programing, and whose achievement would require a more developed technology.

Fig. 6 *The Nimrod* at the Berlin Industrial Show on 6 October 1951. It is a 9 by 12 by 5 feet machine that contains 480 vacuum tubes and executes its program almost independently. (http://www.heise.de/newsticker/meldung/Vor-50-Jahren-fing-alles-an-das-erste-Elektronenhirn-in-Deutschland-51722.html)

As we will see in the next section through several examples, there exists a link between games and their underlying mathematical concepts, and their possible creation through a machine or a computer.

4 The Game of Chess

The game of Chess is considered as the King of all games: International players and Grand Masters are seen as the most intelligent people in the world. Thus, it is not surprising if the first researches led in artificial intelligence started to study the game of Chess (but not only, of course) in order to build a program that could play at a relatively good level (and maybe defeat one day the World Champion). But a few decades before these first programs, an electro-mechanical machine able to play a particular endgame in Chess was designed: *El Ajedrecista*.

4.1 El Ajedresista of Torres y Quevedo

El Ajedrecista (the Chess Player in English, see Fig. 7) was the first serious[6] automaton built in 1911 by the Spanish engineer Leonardo Torres y Quevedo (1915) (1852–1936)[7]. Torres worked on the construction of several devices, which can be divided into two groups: automatons and algebraic machines (Vigneron 1914). An automaton refers to a machine that imitates the appearance and the movements of a man or an animal. Torres explains (Vigneron 1914) that the mechanism has to bear its own source of energy which makes it work (a spring for example) and makes it accomplish some gestures, always the same, without any external influence. Automatons must be capable of discernment and adapt themselves to their environment and to the impressions they receive. Through his Chess Player, "merveille d'ingéniosité" (Vigneron 1914, p 59), Torres proved well that it was possible to create a mechanical machine (in fact, electro-mechanical in this case), which could play Chess.

El Ajedrecista was first exhibited to the public in Paris; it was able to play the special endgame configuration of the white king and rook from any position (held by the machine) against the human black king from any position. *El Ajedrecista*

[6] In 1769, the Hungarian Johann Wolfgang von Kempelen (1734–1804) constructed an automaton Chess player, the Mechanical Turk, able to play at a high level against human opponents. In fact, the Turk was a complete stunt as a human chess master could hide inside the machine and operate the moves [...]. Ajeeb, created by Charles Hooper, succeeded the Turk in 1868, but used the same trick with a hidden player.

[7] "Born in Santa Cruz in the province of Santander in Spain in 1852 and educated as a civil engineer, Torres became director of a major laboratory, president of the Academy of Sciences of Madrid, a member of the French Academy of Sciences, and famous as a prolific and successful inventor. Some of his earliest inventions took the form of mechanical analog calculating devices of impressive originality." (Randell 1982, p 331).

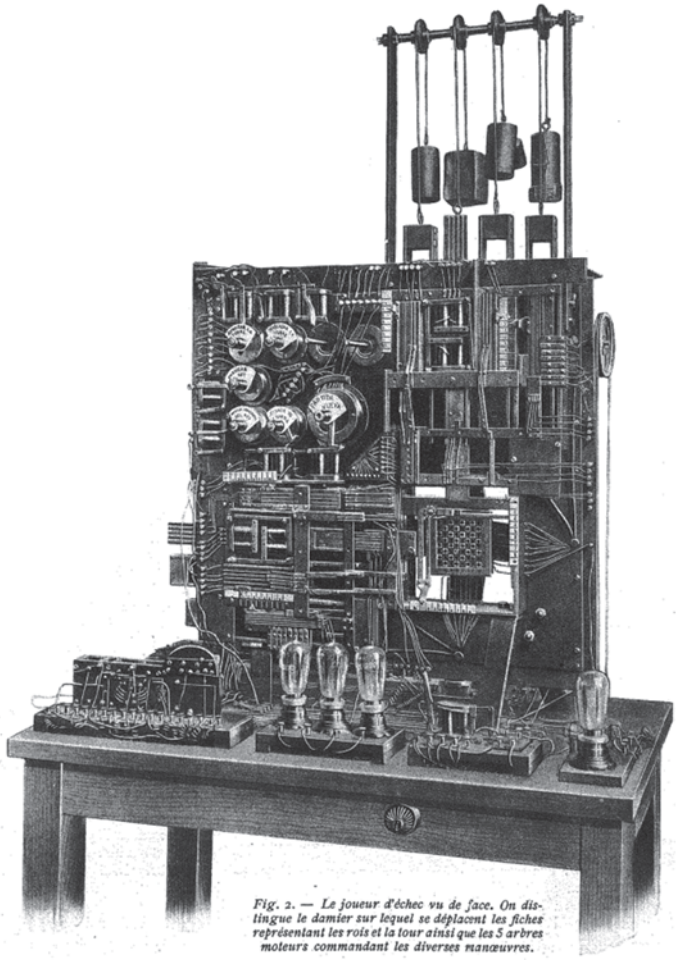

Fig. 7 Front view of the Chess Player (Vigneron 1914, p 57)

runs on a rather simple electro-mechanical system: to every position of the system (*i.e.* of the game) corresponds an electro-magnet. This electro-magnet is activated with electrical connexions when switches are arranged in a given position. Since the switches can be placed in different positions, every move on the chessboard can be obtained by a specific combination of the switches. For example, on Fig. 8, the switch M can take two different positions (A or B) that can activate switches N or N'. If N is activated, it has once again three different positions available (E, F and G), which will activate new switches, and so on until all switches have their own position and form a particular combination.

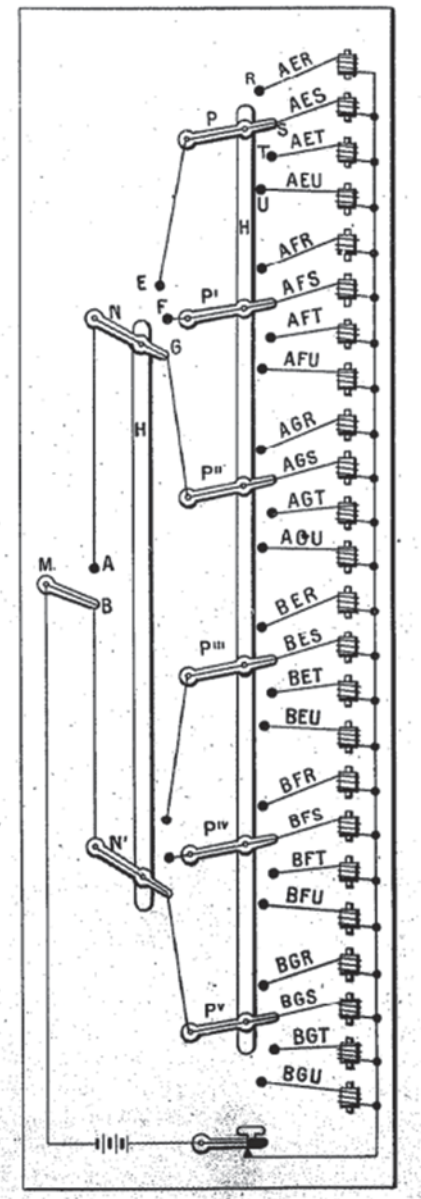

Fig. 3. — Schéma montrant comment on peut déterminer 24 opérations différentes.

Fig. 8 Drawing to show how 24 different operations can be determined in Torres' machine (Vigneron 1914, p 58)

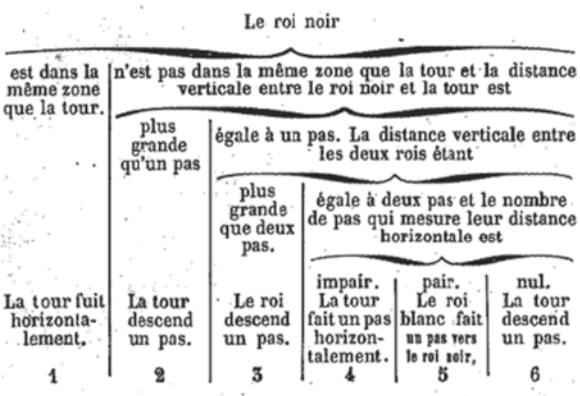

Fig. 9 Set of rules that the automaton has to follow before connecting the switch. (Vigneron 1914, p 60)

Combinations are determined by a set of rules that the automaton has to follow before connecting the switches (Fig. 9).

Torres explains that the number of switches and their associated positions can be increased "as much as we want" (Vigneron 1914, p 59). This means that the number of particular cases can be endlessly increased, which makes the actions of the automaton more complex. Torres considered that there are no essential differences between the simplest machine and a more complicated automaton, since we can provide it the rules it has to follow to execute its moves. The very first version of the automaton used electrical sensing of pieces on the board and a mechanical arm moved the pieces of the machine (Randell 1982, p 332). Some years later, Torres made a second version with magnets underneath the board to move the pieces.

Two paradoxical thoughts arise concerning machines such as *The Nimrod*, *The Nimatron* and *El Ajedrecista*. First, their development and their exhibition did lead to major public events; they remain indeed the first machines ever built to reproduce a situation of a particular game. Even if their popularity was mostly due to the curiosity of the public, their fabrication still symbolises the first achievement of a combinatorial reasoning. And secondly, it is worth noticing that the initial idea was to provide a game and not something more serious. Retrospectively, the reasoning behind these machines is not complicated at all, and it is mainly the advertising made at that moment that contributed to their fame.

4.2 The Beginnings of Artificial Intelligence

Nevertheless these machines did not prevent the development of more serious programs in the 1950s. Some mathematical ideas originate from technologies and these technologies enable their concrete achievement. Creating a program or a machine able to act as the human brain is the main problem of artificial intelligence and the first researches (among others) started with the most noble game of all games: Chess. In 1950, Claude Elwood Shannon (1950) (1916–2001), an electronic engineer and cryptographer, was concerned with the problem of constructing a program for a computer that would enable a computer to play Chess (Shannon 1950). This

led him to lay the foundations of computer Chess programming thanks to the Minimax algorithm, a principle found in every game analysis (not just combinatorial games). This idea was also developed by Alan Mathison Turing (1912–1954) in 1953 as part of his research on artificial intelligence. Here is an explanation of the Minimax's principle.

The Minimax algorithm consists in *mini*mizing the maximum loss in a given position. This principle provides the advantage to evaluate the different positions of the game and to choose the most beneficial one, considering the opponent's moves. The Minimax is a heuristic principle that takes into account the hypothesis that our opponent's aim is to maximise his benefit. In the particular case of combinatorial games, the goal of the two players is clearly opposite: when A wants to maximise his profit, B wants to minimise it (to maximise his own profit).

To apply this algorithm, we first need to represent the game in the form of a tree. Every node of the tree corresponds to a possible position of the game and its branches lead to the positions that can be reached from that node. To evaluate the initial position (called the *root* of the tree, level 0), we need to generate the set of positions reachable from that initial position, we obtain the level 1. We do so as many times as necessary to generate levels 2, 3, …, n. Then we assign a value to each final position, which gives the quality of the position for one of the two players. For example, in Fig. 10 player A (represented by the root of the tree) is about to play and wants to evaluate his position to minimise his loss.

The first step of the Minimax algorithm is to minimise the set of the final positions for every branch. This gives a value to the three parent nodes of the terminal nodes. We find the following minima (Fig. 11).

As player A wants to maximise his profit, the next step is to find the maximum between the three nodes of the level 1 that will correspond to the value of the position and then give the right move to play (Fig. 12).

The Minimax algorithm is based on simple recursive calculations that alternatively minimise or maximise nodes at a given level, actually a simple mathematical process, which is used in every program. If we apply the Minimax algorithm until the terminal positions, *i.e.* if we apply it for the entire game tree, therefore it is equivalent to apply a *backward induction* reasoning.

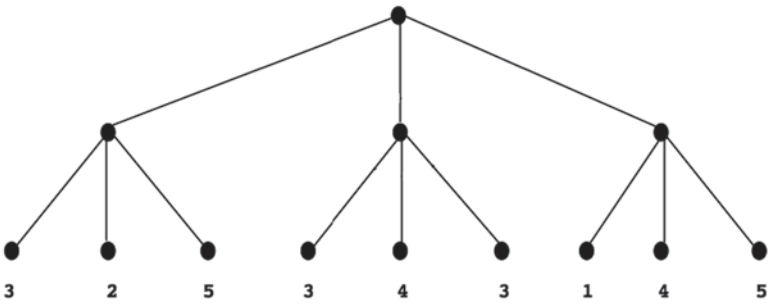

Fig. 10 An evaluation function has evaluated the quality of the final positions from player A's point of view (Alliot and Schiex 1994, p 274)

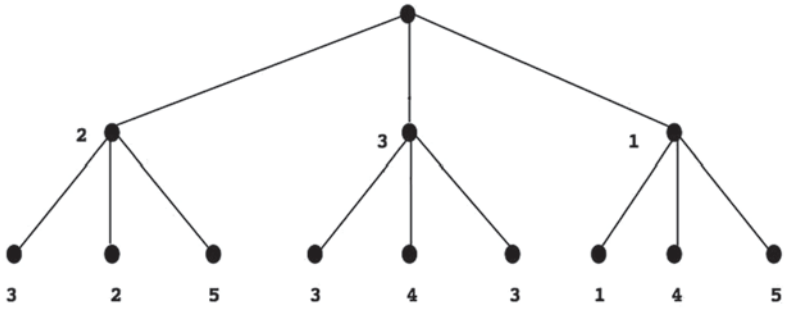

Fig. 11 The minimising step gives the value of the parent nodes of the terminal nodes. They represent player B's strategy that consists of maximising his profit and therefore minimising A's profit. (Alliot and Schiex 1994, p. 274)

Backward induction is used in the resolution of every combinatorial game whenever it is possible. For example, as Chess presents 10^{120} different games, it is impossible to apply backward induction on its game tree, which is also impossible to represent!

Both Shannon and Turing wrote their program based on the Minimax—even if the name did not exist yet—and for machines that did not even exist at that time! It seems (Newell et al. 1958) that there were two other hand simulations between 1951 and 1956 by Frederick Mosteller (1916–2006) and a Russian program but not enough information is available on any of them. Consequently they will not be taken into account in this study. But, as we will see further, the technological advances in computer industry soon gave the opportunity to implement these programs.

In 1956, *Los Alamos Chess* was the first program to run on a computer, MANIAC I, built in 1952 and based on John von Neumann (1903–1957) architecture. *Los Alamos* program is a good example of the system Shannon (1950) described; all alternatives were considered, all continuations were explored to a depth of two moves for each player, the values were determined by a Minimax procedure, and the best alternative was chosen for the move.

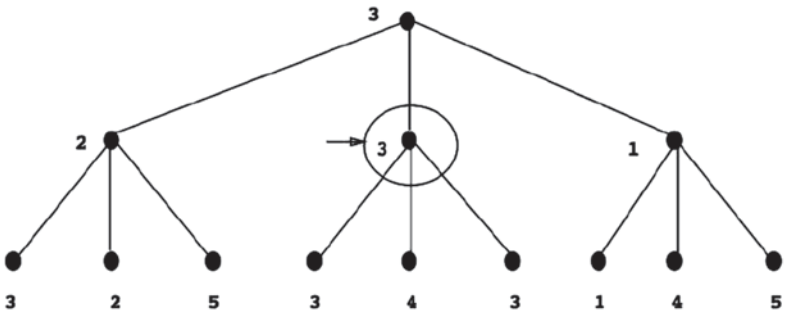

Fig. 12 The value of the initial position is 3 so to maximise his profit, player A will have to choose the second branch (Alliot and Schiex 1994, p. 274)

But the game *Los Alamos* played was a simplified version of Chess on a 6×6 board without Bishops and eliminated special moves such as castling, two-square Pawn moves in the opening and *en passant* captures. These reduced board and rules were chosen in order to carry out the computation within reasonable time limits. In a normal 8×8 game, looking two moves ahead brings about 30^4 that is to say 800,000 continuations to be considered. In the reduced 6×6 game, "only" 160,000 continuations were taken into account. The *Los Alamos Chess* program was developed by Paul Stein and Mark Wells in Los Alamos laboratory. It was able to make a move in about twelve minutes on average and played three games: one against itself, a second against a strong player (it lost), and a third against a beginner (it won).

Two years later a mathematician in the Programming Research Department of IBM, Alex Bernstein, wrote the first program that could play a full game of Chess for the IBM 704, introduced in 1954. The IBM 704 was a very rapid large-scale electronic digital computer, which had performed as many as one billion calculations in a single day in computing the orbit of an artificial satellite (Bernstein and Roberts 1958). Physically, it consisted of several units all under the constant electronic control of the central unit. The principal advanced feature of the 704 was its high-speed magnetic core storage or memory. It replaced the electrostatic or cathode ray tube storage used in the earlier machine systems. A magnetic core was about the size of a pinhead and shaped like a doughnut. Thousands of core were strung on a complex of wires in such a way that several wires passed through the centre of each core. Combinations of electrical pulses on these wires altered the magnetic state of the cores, and a line of cores, some altered, some unaltered, stood for a certain word or number. A word or number stored in the magnetic core memory was available for calculation in 12 millionths of a second. The Type 704 multiplied or divided in 240 μs, or approximately 40,000 operations per second (IBM Archives 1955). Bernstein's program was also in the Shannon tradition (Newell et al. 1958); it could play either Black or White, was capable of playing a complete game of Chess, including moves such as castling, promoting and capture *en passant* and was divided into five parts (1) Input-output, (2) Table generation, (3) Evaluation, (4) Division, and (5) Tree (Bernstein et al. 1958). But an important step was made in the direction of greater sophistication: only a fraction of the legal alternatives and continuations were considered. A set of decision routines was written, which selected a small number (not greater than seven) of strategically good moves. In this way, the program examined 2800 different positions and arrived at the move which, in its estimation, would leave the opponent with the worst possible position. It minimaxed and chose the alternative with the greatest effective value. The time required to the program to make a move was on the average of eight minutes and the machine printed out its move on a sheet of paper. After a mating move or a resignation, the machine printed the score of the game and a paper intended to its opponent with this sentence: "THANK YOU FOR THIS INTERESTING GAME"! (Bernstein and Roberts 1958).

Bernstein's program was the first to give information about radical selectivity, in move generation and analysis.

TABLE 1 Comparison of Current Chess Programs

	Turing	Kister, Stein, Ulam, Walden, Wells (Los Alamos)	Bernstein, Roberts, Arbuckle, Belsky (Bernstein)	Newell, Shaw, Simon (NSS)
Vital statistics				
Date	1951	1956	1957	1958
Board	8 × 8	6 × 6	8 × 8	8 × 8
Computer	Hand simulation	MANIAC-I 11,000 ops./sec	IBM 704 42,000 ops./sec	RAND JOHNNIAC 20,000 ops./sec
Chess program				
Alternatives	All moves	All moves	7 plausible moves Sequence of move generators	Variable Sequence of move generators
Depth of analysis	Until dead (exchanges only)	All moves 2 moves deep	7 plausible moves 2 moves deep	Until dead Each goal generates moves
Static evaluation	Numerical Many factors	Numerical Material, mobility	Numerical Material, mobility Area control King defense	Nonnumerical Vector of values Acceptance by goals
Integration of values	Minimax	Minimax (modified)	Minimax	Minimax
Final choice	Material dominates Otherwise, best value	Best value	Best value	1. First acceptable 2. Double function
Programming				
Language		Machine code	Machine code	IPL-IV, interpretive
Data scheme		Single board No records	Single board Centralized tables Recompute	Single board Decentralized List structure Recompute
Time	Minutes	12 min/move	8 min/move	1–10 hr/move (est.)
Space		600 words	7000 words	Now 6000 words, est. 16,000
Results				
Experience	1 game	3 games (no longer exists)	2 games	0 games Some hand simulation
Description	Loses to weak player Aimless Subtleties of evaluation lost	Beats weak player Equivalent to human with 20 games experience	Passable amateur Blind spots Positional	Good in spots (opening) No aggressive goals yet

Fig. 13 Table of comparison of the different Chess programs in 1958. (Newell et al. 1958, p. 45)

But every increase in sophistication of performance involves an increase in the complexity of the program (Newell et al. 1958). This implies both more program and more computing time per position than with the *Los Alamos* program. Figure 13 compares Chess programs available in 1958 according to their *vital statistics*, their *programming language* or their *results* (Newell et al. 1958, p. 45).

We can observe that Bernstein's program takes 7000 words, the *Los Alamos* program only 600: a factor of about 10. But for time per position, both programs take about the same time to produce the move, 8 and 12 min, respectively. The increase in problem size of the 8 × 8 board over the 6 × 6 board is about five to one; it is approximately cancelled by the increase in speed of the IBM 704 over the MANIAC (also about five to one, counting the increased power of the 704 order code (Newell et al. 1958)). So it can be said that both programs would produce moves in the same 8 × 8 game in the same time. Hence the increase in amount of processing per move in Bernstein's program approximately cancels the gain of 300 to 1 (2800 positions investigated against 800,000 for *Los Alamos*) in selectivity. Newell et al. (1958) deplores this kind of equality between the two programs because selectivity is a very powerful device and speed a very weak one for improving the performance of complex programs. He gives for example the case of both programs that could explore three moves deep instead of two. Then the *Los Alamos* program would take about 1000 times as long as it does to make a move, whereas Bernstein's program would take about 50 times as long.

This aspect emphasises the fact that the time needed for a machine to determine the right move does not only depend on the number of the positions evaluated. The increase of the complexity in the programs also plays an important part in

the time needed. But we can allow programs to become more complex because machines that support them become also more powerful and faster for calculation. And as the performance of the machines increases, researchers become more challenging on the programs… This is how improvements are obtained in Chess programming.

With Bernstein's work we understand that selection in programs is an essential device to increase their level of play. It allows to reduce calculations and to overcome the low speed and power of the machines. A major progress in that direction was made with the development of the Alpha–Beta pruning, a very effective selective cut-off of the Minimax algorithm without loss of information, which is still used in nowadays programs. Here is the principle: on Fig. 11's example, we consider that Minimax algorithm has already analysed the first two branches of the tree and is going to deal the third one. Since the root of the tree is a maximising one (to get its value we maximise values among its children nodes), and one of its children nodes has the value three, we already know that the root will have a value at least equal to three. When analysing the third branch, the first leaf gives the value one. But the parent of this leaf is a minimising one so its value will be less or equal to one whatever the values of the other unexplored leaves are. The latter are therefore uninteresting and will not be analysed by Alpha–Beta. The name of 'Alpha–Beta' and the description we gave are found for the first time in an article of 1963 (Edwards and Hart 1963). The writers underline the importance of ordering the branches of the tree to optimise the Alpha–Beta efficiency: if the level is a maximising one, moves likely to generate high-valued positions must be analysed first and reciprocally if the level is a minimising one. "It turns out at best, that is, in the case of perfect ordering the α-β heuristic can cut a tree's exponential growth rate in half, thus allowing almost twice the search depth for the same effort" (Edwards and Hart 1963, p. 3).

4.3 The Achievement of Technology: Deep Blue's Victory

We saw that most of the problems that arose were connected to the speed of calculation of machines. Therefore researches to reduce the analysis of the game tree progressed, and the Alpha–Beta pruning soon became the idea that prevailed in Chess programming. Since the late 1970s, it had been established that Chess computers became stronger as their hardware speed increased. By 1985 engineers thought that a thousand fold increase in hardware speed might be sufficient to produce a World Champion-class Chess machine (Hsu 2002). But they were soon confronted with problems of material such as the size and the number of the components used in Chess machines. Here is an example through transistors and chips described by Feng-hsiung Hsu (b. 1959) (Hsu 2002). Hsu was specialised on the hardware part of the *Deep Blue* project and before he joined IBM in 1989, he worked on the conception of smaller and faster chips that would permit more calculation.

One of the Chess program's main components is the *move generator*. The move generator generates the Chess moves that the program examines. But to work properly, the move generator needs also to generate unexamined, or not yet searched, moves by the program (Hsu 2002). In the *Belle*[8] design, this second task, performed by the *disable-stack*, required a 64-bit wide memory, one bit for each square of the chessboard. Generally, the program is allowed to look up to 128 plies[9] ahead (so 256 words deep to handle). If we assume that six transistors are needed for every bit of memory, then the number of transistors necessary for the disable-stack alone would be at least 1500 for each square of the chessboard, or about 100,000 transistors for the whole board. And the problem was that in 1985, it was nearly impossible to fit the *Belle* move generator into a single chip; consequently the circuit size was too big. One question engineers had to deal with was the possibility to create a new single chip (instead of the actual 64) to complete *Belle* Chess generator. It led to a re-thinking of the necessity of the disable-stack and after a redefining of its function, only ten transistors on average were used for each square. A 150 to 1 reduction! The same problem was faced with the *evaluation function*[10]: "Could the evaluation function be fitted onto a single chip as well? […] If it is barely possible to fit the move generator onto a single chip, what was the chance of doing the same for a good evaluation function? It doesn't look good, does it?" (Hsu 2002, p. 30). The basic idea to solve this problem was "trade space for time. Chess evaluation functions have spatially repetitive components, and it is possible to use a smaller circuit multiple times to do the same computation" (Hsu 2002, p. 30). Finally, it had been possible to build a single chip Chess move generator and a single chip evaluation function.

This illustrates the important fact that mathematics is a good opportunity for technology to expand and that both cannot be completely detached from each other. In 1996, the year the program *Deep Blue* won one of the six matches against the World Chess Champion Garry Kasparov (born in 1963) in the first ever traditional Chess tournament between man and computer, IBM set a new world record in magnetic data storage density—five billion bits of data per square inch—the equivalent of 312,500 double-spaced typewritten pages in one square inch of disk surface. *Deep Blue* is a combination of special purpose hardware and software with an IBM RISC System/6000, a system capable of examining 200 million moves per second, or 50 billions positions, in the three minutes allocated for a single move in Chess (IBM Archives 2004). Nevertheless, it is a misconception to think that if the computer Chess wins against a human player it is only because it computes faster. The win of the IBM Chess-machine *Deep Blue* against Garry Kasparov in May 1997

[8] Belle is a special purpose Chess machine that was built in the early 1980s by Ken Thompson (b. 1943) and Joe Condon (1935–2012) from the Bell Laboratories. Belle became the first Chess program to play at the US National Master level in 1982 (Hsu 2002).

[9] In computer Chess jargon, a ply designates a move made by one of the two player. In Chess literature a move refers to a move played by White and the answer of Black (or vice-versa), so two plies make a move.

[10] The evaluation function assesses the quality of the positions reached when the Chess machine looks ahead (Hsu 2002).

was the result of many years of researches for the "human" team who had worked on the project. And it is a parallel development between complexity of the programs and efficiency of the machines that permitted this success. Of course, behind all the *Deep Blue* story is hidden the everlasting human desire to create a being at his image. "Writing the technical report and doing the presentation (about the built of a single ship), however, made me realize that I had the basic blueprint to build the Mother of all Chess Machines, a machine that could defeat the World Champion. In other words, I had a chance to pursue one of the oldest holy grails in computer science, and possibly make history" (Hsu 2002, p. 32).

5 Conclusion

The game of Chess is regarded as "one of the most sophisticated of human activities" (Bernstein and Roberts 1958, p. 96). Therefore, it is not surprising if the first attempts to approach human thinking through a machine (problems in simulation of human thinking) were turned toward Chess. It is worth noticing that researches were usually led by mathematicians (Torres, Redheffer, Shannon, Turing, Bernstein, and later the Russian International Grandmaster and computer scientist Mikhail Botvinnik (1911–1995)) and that the first programs were based on basic mathematical concepts (such as the Minimax principle using minimising and maximising functions, the Alpha–Beta pruning using properties of these functions). The evolution of technologies quickly allowed implementation of these programs on machines able to compute faster than humans. And soon, these Chess computer scientists realised that "the level of its chess playing could be considerably improved were this program to be adapted to a bigger and faster machine" (Bernstein 1958, p. 208). So the race for fewer and simpler components to improve the level of the machines started: more powerful hardware would allow more complex search strategy (Marsland and Björnsson 1997, p. 6).

As more and more powerful computers became available, the full pruning capabilities of Alpha–Beta became better known, and so programs permitted more and more calculations. Therefore, improvements of the algorithms are as important as brute-force in the success of Chess programing. Various techniques to improve the move ordering and to make the search more efficient were developed, such as iterative-deepening, use of transposition-tables, and forward pruning (Marsland and Björnsson 1997, p. 8). These various techniques, as well as the establishment of a mathematical model to represent the game through a tree with branches and leaves, are directly connected to the algorithmic aspect of Chess programing. They are quite elementary compare to the deep mathematical theory that arose from Sprague-Grundy theorem, which is actually the mathematical theory of the Nim game and impartial games. Other combinatorial games such as Go or Sprouts use also complex mathematical theory, especially for the endgames. But such a theory for Chess has not emerged; Chess computer scientists successfully performed to represent Chess in an abstract way but, despite the improvement of technologies and comput-

ers, no real mathematical theory of Chess permitting, not only a better implementation of the programs, but also to help a player by giving him another perspective of the game, has been found… yet!

References

Alliot JM, Schiex T (1994) Intelligence artificielle et informatique théorique. Cépaduès, Toulouse
Berlekamp ER, Conway JH, Guy R (2001) Winning ways for your mathematical plays, vol 1, 2nd edn. AK Peters, Wellesley
Bernstein A (1958) A chess playing program for the IBM 704. Chess Review 26(7):208–209
Bernstein A, Roberts MV (1958) Computer V. Chess player. Scientific American 198:96–105
Bernstein A, Roberts MV, Arbuckle T, Belsky MA (1958) A chess playing program for the IBM 704. Proceedings of the 1958 western joint computer conference. Los Angeles, California, pp 157–159
Bouton CL (1901) Nim: a game with a complete mathematical theory. The Annals of Mathematics 2nd Series 3(1/4):35–39
Condon EU (1942) The Nimatron. The American Mathematical Monthly 49(5):330–332
Edwards DJ, Hart TP (1963) The α–β Heuristic. Artificial intelligence project RLE and MIT Computation Centre. Memo 30, 28 Oct 1963, revised version of The Tree Prune (TP) Algorithm, AI, Memo 30, 4 Dec 1961
Eiss HE (1988) Dictionary of mathematical games, puzzles, and amusements. Greenwood, New York
Gardner M (1959) Hexaflegons and other mathematical diversions The first scientific American book of puzzles & games. The University of Chicago Press, Chicago
Hsu F (2002) Behind deep blue: building the computer that defeated the world chess champion. Princeton University Press, Princeton
IBM Archives (1955) 704 data processing system. Manual of IBM electronic data processing machines. http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP704.html. Accessed 21 Dec 2014
IBM Archives (2004) IBM highlights, 1996–1999. http://www-03.ibm.com/ibm/history/documents/pdf/1996–1999.pdf. Accessed 21 Dec 2014
Marsland TA, Björnsson Y (1997) From Minimax to Manhattan. In: Deep Blue versus Kasparov: The significance for artificial intelligence. AAAI Workshop, 1997, pp 5–18
Moore EH (1910) On the foundations of mathematics. Bulletin of the American Mathematical Society 9(8):402–424
Newell A, Shaw JC, Simon HA (1958) Chess-playing programs and the problem of complexity. IBM Journal Research and Development 2:320–355
Randell B (1982) From analytical engine to electronic digital computer: the contributions of Ludgate, Torres, and Bush. Annals of the History of Computing 4(4):327–341
Redheffer RM (1948) A machine for playing the game Nim. The American Mathematical Monthly 55(6):343–349
Shannon CE (1950) Programming a computer for playing chess. Philosophical Magazine Series 7 41(314):256–275
Torres L (1915) Essais sur l'automatique. Revue générale des sciences pures et appliquées. Tome 26:601–611
US Patent (1940) Condon et al., Machine to play game of Nim. US Patent 2,215,544 24 September 1940, p 2
Vigneron H (1914) Les automates. Nature 2142:56–61