A-Games: using game-like representation for representing finite automata

Cleyton Slaviero[1], **Edward Hermann Haeusler**[2]

[1]Instituto de Ciências Exatas e Naturais
Universidade Federal de Rondonópolis (UFR)
Rondonópolis – MT – Brazil

[2]Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro – RJ – Brazil

slaviero@ufr.edu.br,hermann@inf.puc-rio.br

Abstract. *Non-determinism in automata theory allows us to model situations where given an input, one or more outputs are possible. Although this decision regarding which state to chose could be random, there are contexts where this decision is not random, for instance when modeling real life situations. Using game theory, we propose the representation of automata as a game of two players. This game is defined for languages of finite size. We characterize that this representation is suited for both deterministic and non-deterministic automata, and relate the former with perfect information games, and the latter with imperfect information games. We argue that this could help us in explaining concurrency in programming, for instance in novice programming environments.*

1. Introduction

Automata theory studies abstract computing devices [1]. These devices, or machines, have been used in many contexts, and allow us to model and understand interactions in digital circuits; in lexical analysis; software for analyzing large data sets; and software for verifying systems such as communication protocols. For simple problems, such as turning a switch on or off, it is straightforward to define the set of possible paths of the automaton, or the language that it recognizes. However, when modeling more complex systems and using more complex automata, with non-deterministic features for instance, one question arises: how to "decide" if a given transition is chosen over another, given the same input? These kinds of decision problems may arise, for instance, in concurrent systems, which can be modeled by automata such as parallel automata or even Petri Nets [2, 3, 4], or in simple games created by novice programmers. In a previous work [5] we found that the novice programming environments present different approaches to concurrency. Learning concurrency and understanding situations such as synchronization and order-of-execution is a challenge [6] and many bugs regarding concurrency arise when analyzing programs [7]. To this end, our question arises: how could we better understand and tackle the understanding of concurrency? The famous seminal work from Morgenstern and von Neumann [8] defined an area in economic theory called Game Theory. Since then, researchers have been investigating the connection between computer science and game theory [9, 10]. Game theory studies decision problems, in which two (or more) players need to decide, given their interests, beliefs and knowledge about the other(s), what actions to take. The connection between game theory and computer science has been explored to great extent [9, 10]. Shoham [10] presents a comparison between a work from Kalai in 1995 and current trends in game theory and computer science. From the last two foci the author mentions, "logics of knowledge and belief, and other logical aspects of games" remains a key area to approach game theory and computer science. In this research, we attempt to discuss this relationship between game theory and computer science via automata theory. This is not new: the seminal works from Reif [11, 12] discuss this relationship by means of describing complexity in computing a game using Turing machines. Many subsequent works have been investigating these kinds of relationship, such as the works on algorithmic game theory [13, 14]. In these works, automata are employed to discuss the complexity of strategies in games. In our context, we want to do the opposite: discuss automata and the characteristics of systems they represent by using a game-theoretical approach. Our goal is to investigate how game theory could support the characterization of non-determinism in terms of rational behavior. To this extent, we propose to represent automata as games. In this work we start by showing how games and finite automata could be related. Furthermore, we show that finite deterministic automata are equivalent to perfect information games, whilst finite non-deterministic automata are equivalent to imperfect information games. We finish this paper by showing further works that could be explored from these results. We expect that these new definitions help us in exploring concurrency problems and proposing new ways to explore and solve them.

2. Automata Theory

Automata theory (AT) studies abstract computing devices or "machines" [1]. Automata can be used to model, for instance, in lexical analysis; text corpus analysis; and systems verification. Formally, a Finite State Machine, or Finite Deterministic Automata (FDA), is defined by D = $\langle Q_D, \Sigma, \delta_D, q_0, F_D \rangle$, where:

- Q_D is a non-empty finite set (of states);
- Σ is a finite (non-empty) set of symbols (the alphabet)
- q_0 is the initial state
- F_D is the finite set of final states (or accepting states)
- $\delta_D : Q_D \times \Sigma \to Q_D$ is the possibly partial transition function taking as argument a state and an input and returning a state.

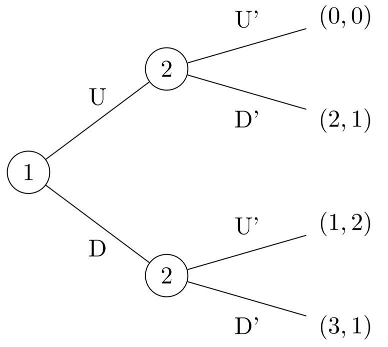
We can also define $\hat{\delta}(q, w)$, $q \in Q_D$ and $w \in \Sigma^*$, and returns $q' \in Q_D$ as the state that automata reaches when it runs over w [1]. From $\hat{\delta}$ we are able to define L(A), which is the language accepted by A as $\{w|\hat{\delta}(q_0, w) \in F_D\}$. [1].

To define a non-deterministic finite automaton (NFA), $\delta : Q_D \times \Sigma \to \mathcal{P}(Q_D)$ is a function which takes a state and an input and returns a set of states.

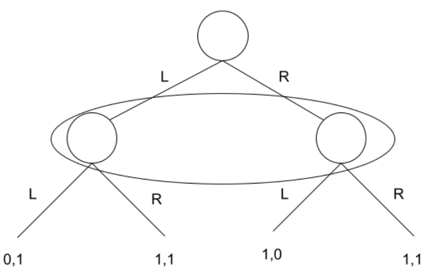
3. Game Theory

Informally defined, a perfect information game is a game where each player knows the move other player's chose when playing the game. More formally, a (strategic) game is a 3-uple $\langle P, A_i, \succsim_i \rangle$ where [15]:

- P is a set of players



(a) Extensive form.



(b) Imperfect Information Game

Figure 1. Examples of games.

- A_i is a set of moves of player i
- $\succcurlyeq_i$ is a preference relation of player i.

A famous example of a game is the Prisoner's Dilemma, in which two prisoners must choose to confess or deflect from a given accusation. In either choice they could do, there is a better or worse payoff considering the other prisoners' decision. In Figure 1a we present this game in extensive form, one type of representation for games. In this picture, preferences are described as numerical values. This follows as a consequence of the utility theory from Von Neumann and Morgenstern [8], in which $x \succcurlyeq y$ iff $U(x) \geqslant U(y)$.

Another class of games is the one where a given player has no information about previous moves from other player(s) [15], which we call imperfect information game. In this class of games, an information set groups possible outcomes the player might be at, after a move from previous players. The current player has no knowledge of which move the previous players performed. Figure 1b show an example of an imperfect information game where the players must choose between left and right during their moves. However, after player one chooses either move, player two is not aware of player's 1 choice.

4. Automata as Games

In order to describe automata as games, we start by providing a game representation for finite deterministic automata. In this section, we describe the general idea and provide examples of automata and the equivalent game.

Definition 4.1 *(A-Game) Let A be a finite automaton $\langle Q_A, \Sigma, \delta_A, q_0, F_A \rangle$ and let G_A be an extensive game with two players, P_I and P_{II} . In this game, P_I plays states from Q_A and P_{II} plays characters from Σ_A^*.*

The game is played as follows. P_I starts by playing q_0, the initial state. In order to proceed, P_{II} must play any character $s \in \Sigma^*$. The game ends when P_I plays a state $q \in F_A$ and P_{II} has no more moves. This will be later defined as a winning strategy. However, we must first define the concept of strategy for these players.

Definition 4.2 *The strategy for each player is the following: if P_I plays $q \in \Sigma_A$ then P_{II} must play some s, such that $\delta(q, s) \neq \emptyset$, or else P_{II} loses.*

It happens that if A is deterministic, then G_A is equivalent to a perfect information game. However, if A as non-deterministic, then $\delta(q, s) = B \subseteq Q_A$, and thus we would

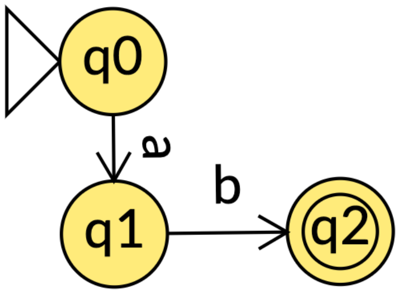


Figure 2. A finite deterministic automaton.

have an information set around P_{II} 's next move after P_I plays $q“ \in B$. This allows us to infer that δ defines the information sets for P_{II} . Consider that P_I plays $q“ \in B$. Then, P_I must play $s“ \in \Sigma$ such that $\delta(q, s‘)$ is defined and $q \in B$.

In this game, we can also define the strategies of each player. According to the definition from [15], the strategy of a player assigns an action chosen by the player for *every* history in which it is his turn to move. The history of a player is defined by the set of actions he might choose during the game play. For P_I this is equivalent to all combination of states he might chose to confront P_{II} ; regarding P_{II} , these histories is any word $w \in \Sigma^*{}_A$.

Definition 4.3 *Let A be an automaton and $w \in \Sigma^*{}_A$. A strategy for P_{II} defined by w is such that P_{II} chooses symbols from w sequentially, apart from what P_I chooses.*

Another important characteristic of the a-Game is that if we consider a finite automaton, we must consider that the accepted words are those of length n, at most. Thus, in an a-Game we deal with any $L(A)^n$, which is the set of words of length n.

Definition 4.4 *Let $S_I(G_A)^n$ be the set of all games on G_A where P_{II} plays according to the strategy given by Σ^n_A, or all words of size n.*

Informally, an a-Game is a game where player one is the *state writer* and the player two is the *character writer.*. In other words, player one plays any state $q \in Q$, and player two plays any character $s \in \Sigma$. In Figure 3, we show an example of an a-Game. Notice it has the size of the accepted word of the automata in 1b, where there are only three states and two possible transitions. q_2 is the final state, and q_0 is the initial state. The final move from 2, in this example, can be a, b or λ, a flag to denote that P_{II} has no more characters from $\Sigma*$.

4.1. Winning strategies and language acceptance

In order to relate the game and the automata, we must show that any winning strategies in the game is equivalent to the language accepted by A. In Definition 4.2, we unveil the concept of (general) strategy for each player. Also, in Definition 4.3, we describe a strategy for P_{II} and a word w. However, we are interested in those where P_{II} wins over P_I , which are the winning strategies of P_{II} .

Definition 4.5 *A winning strategy in the a-Game for P_{II} is a strategy w of length n, where P_I has no valid move to play and last move is $q_f \in F_A$, or the set of final states of A.*

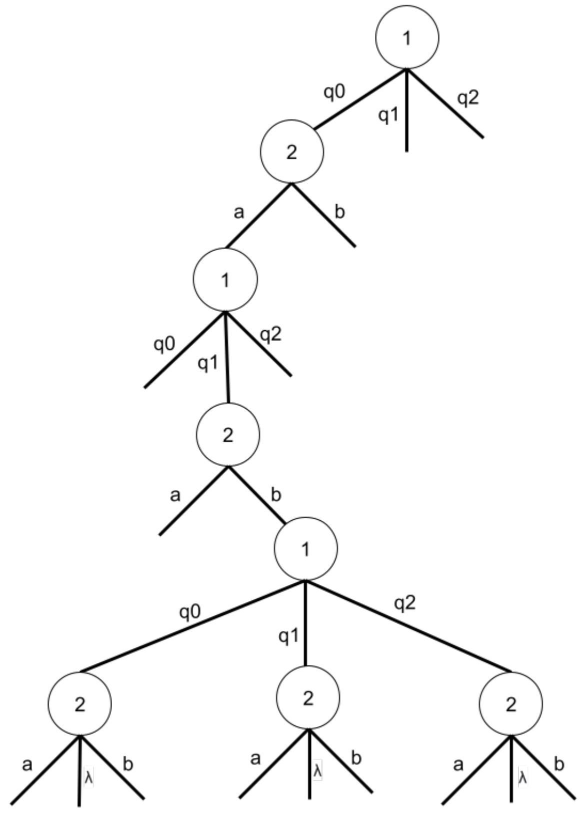


Figure 3. Example of an a-Game from the automata in Figure 2.

This definition allow us to talk about the equivalence between the winning strategies in the a-Game and the language acceptance of automaton A.

Definition 4.6 *Let $S(G_A)^n$ be the wining strategies of G_A^n and the words accepted by the automata A, $L(A)^n$.*

In essence, we need to show that given a winning strategy $s \in S(G_A)^n$, there is an application of $\hat{\delta}$ over a word w of length n, where we end in a final state of the automaton. In other words, w is recognizable by A, or $w \in L(A)^n$.

Theorem 4.1 **($S(G_A)^n = L(A)^n$)**

Proof. ($S(G_A)^n \subseteq L(A)^n$) By induction on the size of the strategy. (Basis) The empty strategy. In this case, P_I plays the initial state and the second player plays empty. This means the initial state of A is a final state. Thus, the empty string belongs to $L(A)^n$. (Inductive step) Assume that $S(G_A)^n = L(A)^n$, where n is the length of a string/strategy. Let $(a_I; a_{II}) = s \in S(G_A)^n$ where a_I and a_{II} are the preceding actions from P_I and P_{II} respectively. Since $S(G_A)^n = L(A)^n$, there is a $\hat{\delta}(q, w)$ with $w \in L(A)^n$. Then, let s' be a strategy of length $n + 1$ in the form $(a_I; a_{II}; c)$, $c \in \Sigma$. If $s' \in S(G_A)^n + 1$, then by inductive hypothesis, $\delta(q, \hat{w}c) \in L(A)^{n+1}$. ($S(G_A) = L(A)$.)
($L(A)^n \subseteq S(G_A)^n$) By induction in the size of the string. (Basis) A string $w \in L(A)^1$. In this case, P_I has one move, and P_{II} also plays, which leads to a winning strategy for P_{II} . (Inductive step) Assume that $L(A)^n = S(G_A)^n$, where n is the length of the string. For $L(A)^{n+1}$, we can represent it in the form $w = aw'$. Then, $a \in L(A)$ and has a winning

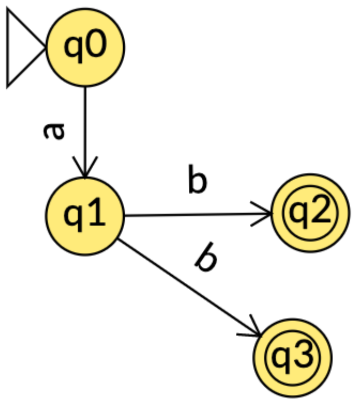


Figure 4. A simple NFA.

strategy, by the inductive step. Since $w' \in L(A) \mid_n$, then $s'_w \in S(G_A)$. $w' \in \Sigma_{L(A)}$ Since $w \in L(A)$ $s \in G(A)$.

QED

5. Ai-Game: A non-deterministic finite automaton Game with imperfection information

We also consider here the case of non-deterministic finite automata. In order to represent it, we observed that the non-determinism from the automata could be equivalent to the concept of information sets available in the imperfect information a-Game concepts. We then present in this section a description of imperfect information a-Game, based on a non-deterministic finite automaton. An a-Game with imperfect information is a game representing a NFA (non-deterministic finite automaton). It can be defined formally as follows:

- A set P with P_I playing $w \in \Sigma$ and P_{II} playing $q \in Q$
- A set of moves $S = \Sigma \times Q$
- A set I of information sets, where each information set is defined as a move from a state in the automaton where there is non-determinism, i.e., $\sigma(q, s) = \{q_1, ..., q_n\}, qi \in Q$
- $\succeq_i$: is a **preference relation** between any given transitions σ_{A_i} and $\sigma_{A_{ii}}$ where $\sigma_{A_i} \succeq \sigma_{A_{ii}}$ iff $d(\sigma_{A_i}) \leq d(\sigma_{A_i ii})$, with $d : Q \to \mathbb{N}$ as a function that given a transition, returns the smaller path from the output state to the initial state of the automata.

5.1. Example of an Ai-Game

Let us show how we can represent a NFA as an imperfect information a-Game. Figure 4 shows a simple non-deterministic automaton, whilst 5 shows the game-like representation of the game. Notice that, for each decision that leads to a non-accepting state, we place an x. The information set, in this case, is placed in the last move. In this case, that last moves that lead to a winning strategy are those that show “ok” at the end. We use λ, a flag to represent that the move where P_I is at a final state.

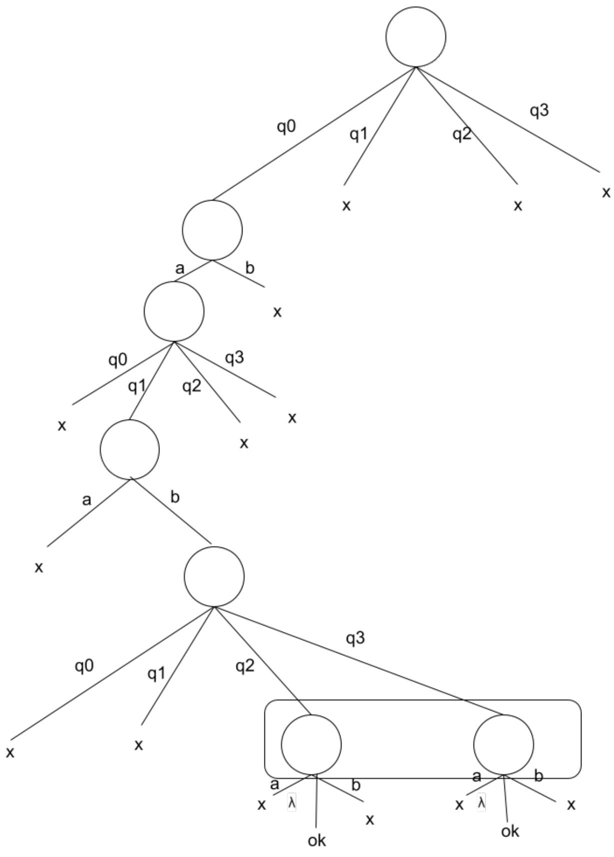


Figure 5. Representing a NFA as a game.

5.2. Winning strategy for Ai-Game NFA

As we discussed before, we also need to demonstrate the equivalence between the results shown in a NFDA and an imperfect information a-game. Intuitively, we find that the deterministic path in the imperfect information A-game is similar to the first A-game we presented. The main difference is that when reaching an information set/non-deterministic transition, P_{II} has no information regarding P_I previous move. However, the same theorem applied to an a-Game, as defined in 4.1, also applies to an Ai-Game.

6. Remarks and future works

Game theory and computer science have a common history of relationship via distinct approaches. In this paper, we attempt to discuss implications of bringing together these two theories, specially, regarding adding the concepts of knowledge and rationality [15] to (non-deterministic) automata. Specially, we are interested whether the concept of rationality can be employed to describe conflict situations in problems modeled by automata. To this end, we provided two distinct representation for different types of games and automata. We aim to explore these equivalences even further, initially exploring the relationship of more general games with more expressive kinds of automaton; and also discuss how rationality and some of its derived properties can be related to properties in these automata. Further work also aims to explore the subset construction algorithm, that allows us to convert a NFA to an FDA. We aim to explore if this relationship could be transposed to the games we created and if game theory could help us understand the pragmatic perspective of this conversion, for instance analyzing the concept of dominant/dominated equilibrium strategies, and other types of equilibrium, such as Nash Equilibrium, and how it could be related to the language acceptance of (finite) automata.

References

[1] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

[2] David Park. Concurrency and Automata on Infinite Sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, London, UK, UK, 1981. Springer-Verlag.

[3] Lutz Priese. Automata and Concurrency. *Theoretical Computer Science*, 25, 1983.

[4] Manfred Droste and R. M. Shortt. From Petri nets to automata with concurrency. *Applied Categorical Structures*, 10(2):173–191, 2002.

[5] Cleyton Slaviero and Edward Hermann Haeusler. Exploring Concurrency on Computational Thinking Tools. In *Anais do XIV Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais*, Salvador-BA, 2015.

[6] Yifat Ben-David Kolikant. Learning concurrency: Evolution of students' understanding of synchronization. *International Journal of Human Computer Studies*, 60(2):243–268, 2004.

[7] Shan Lu, Soyeon Park, Eunsoo Seo, and Yuanyuan Zhou. Learning from Mistakes - A Comprehensive Study on Real World Concurrency Bug Characteristics. In *ASPLOS'08*, page 340. Association for Computing Machinery, 2008.

[8] John Von Neumann and Oskar Morgenstern. *Theory of Games And Economic Behavior*. Princeton University Press, 60 edition, 1944.

[9] Joseph Y. Halpern. A computer scientist looks at game theory. *Games and Economic Behavior*, 45(1):114–131, 2003.

[10] Yoav Shoham. Computer science and game theory. *Communications of the ACM*, 51(8):10, 8 2008.

[11] John H. Reif. Universal games of Incomplete Information. In *11th Annual Symposium on Theory of Computing*, pages 288–308, Atlanta, GA, 1979.

[12] John H Reif. The Complexity of Two Player Games of Incomplete Information, 1984.

[13] Tim Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53(7):78, 2010.

[14] Noam. Nisan. *Algorithmic game theory*. Cambridge University Press, 2007.

[15] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory.* Number 1. MIT Press Books, 1994.