NON-DETERMINISM IN CS HIGH-SCHOOL CURRICULA

Michal Armoni[1] and Judith Gal-Ezer[2]

***Abstract** - One of the units in the relatively new high school CS curriculum which is being implemented in Israel is a theoretical unit on computational models. It includes deterministic and non-deterministic finite automata, regular and non-regular languages, closure properties of regular languages, pushdown automata, closure properties of context free languages, Turing machines, the Church-Turing thesis and the halting problem. This paper focuses on part of a study we conducted on the unit, dealing with the topic of non-determinism of finite automata. One of the aspects dealt with was how students perceived non-determinism. 339 students were given a relatively complicated regular language, and asked to construct a finite automaton that accepts this language. We found that many students did not choose the easiest way to solve the problem: Many students preferred to construct a deterministic automaton, even though constructing a non-deterministic automaton for the language is much simpler. We analyze and categorize the students' solutions, thus shedding some light on their perception of the abstract concept of non-determinism.*

Index Terms – Computational model, Deterministic fnite automata, Non-determinism, Non-deterministic finite automata.

COMPUTATIONAL MODELS IN THE HIGH SCHOOL CS CURRICULUM IN ISRAEL

A relatively new high school CS curriculum is being implemented in Israel [1, 2] which has two different versions, a three-unit version and a five-unit version. The fifth unit is an elective component, and one of the alternatives is a theoretical unit on computational models (CM), for which a textbook and a teachers' guide were written [3]. This unit was developed by a team chaired by the first author, in consultation with the second. The unit is planned for 90 hours, and taught during one school year. It has three parts. The first and largest part (about 50 hours) deals with finite automata. This part introduces various models of finite automata, and discusses the equivalence of these models. It also includes a discussion of the limits of computation of finite automata (that is, the existence of non-regular languages) and provides proofs of some closure properties of regular languages.

The second part (about 25 hours) focuses on the pushdown automata model. After introducing the new model, pushdown automata are proved to be stronger than finite automata. As in the first part, this part also discusses the computational limits of the model, and the closure properties of the family of languages accepted by this model: context-free languages.

The third and last part (about 15 hours) is dedicated to the Turing Machine model. As was the case for the previous models, after introducing the new model, there is a discussion of its computational power (that is, its equivalence to a computer) and its computational limits (demonstrated by proving the non-computability of the halting problem).

Most of the topics introduced in the CM unit are not usually covered in high school CS curricula; some of the technical issues that relate to constructing automata are sometimes touched upon but without discussing any theoretical aspects. The ACM high school computer science curriculum [4], for example, includes very few references to some of these, and then only as optional topics. However, most academic curricula [5, 6, 7, 8] recognize these issues as fundamental to computer science. Therefore, the designers of our high school CS curriculum decided that it was important to expose high school students to these issues to enable them to become familiar with some of the theoretical aspects of computer science.

Non-determinism in the CM unit

The concept of non-determinism is introduced in the fourth chapter of the CM unit. The non-deterministic finite automaton (NFA) model is usually defined as a straightforward version of the definition of the deterministic finite automaton (DFA) [9]. In a DFA, the transition function maps each state and input letter to a single state, while in an NFA, the transition function maps each state and input letter to a set of states (which can also be empty). To illustrate, figures 1 and 2 show two simple automata, both accepting the language that contains all words over the alphabet {a, b, c} which end with the string "bc". The first automaton (Figure 1) is deterministic, while the second one (Figure 2) is non-deterministic.

[1] Michal Armoni, The Open University of Israel and Tel-Aviv University, 16 Klausner St., Ramat-Aviv, P.O. Box 39328, Tel-Aviv, 61392, Israel, michal@openu.ac.il
[2] Judith Gal-Ezer, The Open University of Israel, 16 Klausner St., Ramat-Aviv, P.O. Box 39328, Tel-Aviv, 61392, Israel, galezer@openu.ac.il

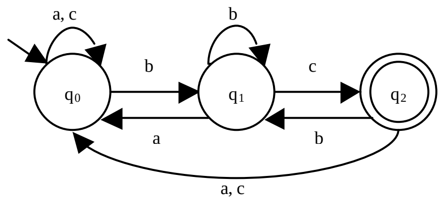

FIGURE 1
AN EXAMPLE OF A DFA

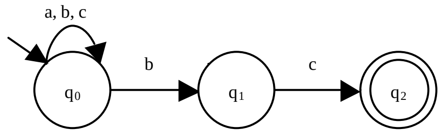

FIGURE 2
AN EXAMPLE OF AN NFA

For didactic reasons, the introduction of NFA in the CM unit is preceded by the introduction of another model: the non-complete deterministic finite automaton (NCDFA). In this model, the transition function maps each state and input letter to a single state or to an empty set of states. Figure 3 shows an example of an NCDFA that accepts the language of all the words over the alphabet {a, b, c} that begin with the string "bc".

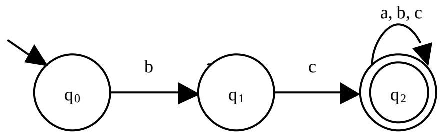

FIGURE 3
AN EXAMPLE OF AN NCDFA

Thus, the first difference between a DFA and an NFA, which permits omitting transitions while preserving the deterministic nature of the model, is shown in the definition of the NCDFA; while the second difference, which permits non-determinism, is expressed only in the definition of the next model taught, the NFA. The abstract concept of non-determinism is thus isolated, and is introduced by itself. The addition of the NCDFA model also enriches the variety of models introduced in the CM unit, and enables practicing comparison of models.

For the same didactic reasons, the definition of NFA in the CM unit does not permit ε-transitions (which are transitions that can be made without reading any input symbol). The concept of ε-transitions is even more abstract and subtle, and is not really needed for the definition of a non-deterministic model (since the resulting models are equivalent).

The unit explains the motivation for introducing the NFA in three ways:

- Practicing the "theoretical game" which characterizes the theoretical study in CS. That is, after a certain mathematical abstraction is defined (in this case the definition of DFA), it is interesting to check the theoretical results of generalizing this definition in various ways: whether the resulting models will be equivalent to the original one, stronger or weaker.
- A few examples, given in the chapter, demonstrate that for certain formal languages, constructing an NFA is simpler and more natural than constructing a DFA.
- By using the non-deterministic model, additional closure properties of regular languages can be proven.

The second part of the fourth chapter focuses on practicing constructions of NFAs and NCDFAs, and studying the properties of the new models.

THE STUDY

The work described in this paper is part of a wider research which examined the correlation between achievements of students studying the CM unit, and other factors such as the student's previous computer-related background (not necessarily computer science), the grade level (11th or 12th), and the level on which the student studies mathematics (which can be 3, 4 or 5 units). We also checked to what extent the students use reductions when solving questions related to computational models, and their perception of non-determinism. This paper focuses on the latter.

Developing the CM unit involved a three-year long experiment, during which the unit was taught in selected schools, under the close supervision of the developing team. The majority of the study population includes students who took the CM unit in 1997-98, the third year of the experiment. The rest are students who learned the unit in 2000-01.

In the third year of the experiment, all the teachers who taught the CM unit were asked to administer a background questionnaire to their students at the beginning of the year. The questionnaire included the students' grade level, previous computer-related background and level of mathematics studied. During the year, the teachers were asked to include a number of questions provided by the developers on exams and to send the students' answers to the developers. At the end of the year they were asked to send the developers the students' answers on the final exam. The same data was collected from the classes which studied the CM unit in 2000-01.

The research instrument

One of the exam questions provided to the teachers tested the material in chapter 4 of the CM unit. The teachers were asked to include the question on an exam given after they finished teaching the chapter. The rest of the questions were written by the teachers. This question, reproduced below, served as our main research instrument for collecting the data reported in this paper:

> Design an automaton that accepts the language over the alphabet {a, b, c}, that contains exactly the words for which at least one of the following conditions holds:
> 1. The word ends with the string "bc".
> 2. The word consists of two parts: The first part contains the string "ba", and the second part contains the string "ab".

There are a number of ways to answer this question, which are listed below in brief:

- Directly constructing a DFA for this language. This is a relatively complicated DFA, containing 8 states and 24 transitions.
- Directly constructing an NFA for this language. If the student understands the non-deterministic model well, the automaton is not very difficult to construct; it contains 7 states and 9 transitions.
- Decompose the language into two sublanguages (corresponding to conditions 1 and 2), construct a DFA for each, and use a non-deterministic union construction to obtain an NFA that accepts the union of the two sublanguages.
- Decompose the language into two sublanguages (corresponding to conditions 1 and 2), construct an NFA for each, and use a non-deterministic union construction to obtain an NFA that accepts the union of the two sublanguages.
- Decompose the language into two sublanguages (corresponding to conditions 1 and 2), construct a DFA for each, and use a Cartesian-product construction to obtain a DFA that accepts the union of the two sublanguages. This construction, if performed gradually, can result in a DFA with 10 states and 30 transitions.
- Decompose the language into three sublanguages (corresponding to condition 1 and the two sub-conditions of condition 2), construct a DFA for each, and use a non-deterministic concatenation construction and then a non-deterministic union construction to obtain an NFA that accepts the union of the first sublanguage with the concatenation of the other two.
- Decompose the language into three sublanguages (corresponding to condition 1 and the two sub-conditions of condition 2), construct an NFA for each, and use a non-deterministic concatenation construction and then a non-deterministic union construction to obtain an NFA that accepts the union of the first sublanguage with the concatenation of the other two.

All of these solutions represent correct solution patterns, though the resulting solution may be incorrect if the student erred in one or more of the stages of the solution (in constructing the automata, in identifying the correct regular operation, etc.). There is one more way to solve the problem; this one is necessarily wrong:

- Decompose the language into three sublanguages (corresponding to condition 1 and the two sub-conditions of condition 2), construct a DFA for each, and use a non-deterministic concatenation construction to obtain an NFA which accepts the concatenation of two of the sublanguages and then use a Cartesian-product construction to obtain an NFA that accepts the union of the first sublanguage with the concatenation of the other two.

This solution is wrong since the Cartesian-product construction was defined and proved for DFAs and not for NFAs, and the concatenation construction can result in an NFA, even if the basic automata were deterministic.

QUANTITATIVE RESULTS

A total of 11 teachers in 9 schools teaching 339 students in 17 classes submitted their students' answers to the question described above.

For the purposes of this paper, we decided to ignore the issue of correctness and focus only on the way in which the student chose to solve the problem. Obviously there is a connection between the method of solving the problem and the correctness of the solution. For example, if the solution involves constructing complicated automata, then there is a good chance that the student will make errors in the constructions. However, we were mainly interested in finding out whether the students chose to solve the problem using the non-deterministic model. Since the fourth chapter of the CM unit emphasizes and demonstrates the relative ease of the construction process in the non-deterministic model, as compared to the deterministic model, we reasoned that if, in spite of that, the students preferred to use the deterministic model, this may indicate that they did not fully understand the non-deterministic model.

Two main factors are involved in the process of solving the problem: the reduction of the problem into sub-problems and the use of non-determinism. Though these two factors may seem orthogonal to each other, they are not fully independent. For example, if the student chooses to decompose the language into two or three sublanguages, the resulting sublanguages will be quite simple, and therefore constructing a DFA for each will not be overly complicated. However, in this paper we limit ourselves to the factor of non-determinism. Decomposition was part of our wider study, and will be discussed elsewhere.

In relation to the use of non-determinism, the students' answers can be divided into 5 groups:

- Fully deterministic solutions
- Solutions in which the students used decomposition to two or three sublanguages. The automata built for these sublanguages were fully deterministic, and only the use of construction algorithms introduced non-determinism

into the process. We categorized this as a deterministically-based solution since when independent thinking was required, the student used the deterministic model.

- Almost deterministic solutions, with only few local non-deterministic behaviors.
- Almost non-deterministic solutions, with only a few instances in which the student ignored the freedom of the non-deterministic model and used redundant transitions.
- Fully non-deterministic solutions.

When categorizing the students' answers, we found no solutions which could be defined as "equally deterministic and non-deterministic". This is not surprising. It is reasonable to assume that if students do not understand the non-deterministic mechanism, they will not use it (partially or at all), whereas if they understand the mechanism and its advantages, they will try to utilize it as much as possible. The distribution of the various types of solutions is shown in Figure 4

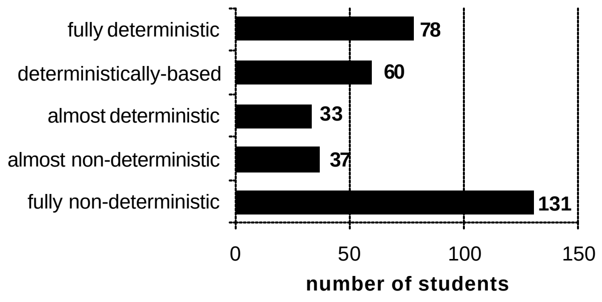

FIGURE 4
SOLUTIONS BY TYPE

Figure 5 is a version of Figure 4, but combines the three deterministic or almost deterministic columns, and the two almost non-deterministic and non-deterministic columns.

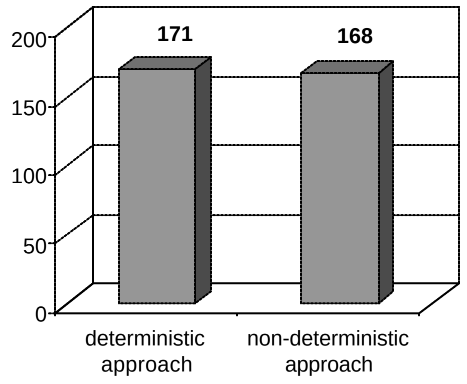

FIGURE 5
DETERMINISTIC VS. NON-DETERMINISTIC APPROACH

About half of the students solved this question deterministically, or almost deterministically. No significant differences were found for grade (11th and 12th) or level of mathematics. This is surprising since in a study about the perception of the concept of efficiency [10], Gal-Ezer and Zur found a significant difference among 10th and 11th graders. However, this may have been the case because the students were not only in different grades, but also at a different stage of their CS studies.

When we considered the data for each teacher separately, we found that for about half of the teachers (5 of 11), the ratio between students using the deterministic approach and those using the non-deterministic approach was about 50-50. For 3 teachers the ratio is about 60%, 70% and 85%, in favor of the deterministic approach, while for the other three teachers, the ratio is about 70%, 85% and 90% in favor of the non-deterministic approach. Thus, it seems reasonable to hypothesize that the teacher factor is significant. If the teacher emphasizes the non-deterministic model, even more than it is emphasized in the textbook, and demonstrates its advantages (for example, by using the non-deterministic model whenever he/she constructs an automaton, both when teaching chapter 4 and thereafter), this may affect the students' tendency to use the model.

QUALITATIVE RESULTS

In analyzing the students' solutions, we recognized four patterns which seem to indicate the existence of a problem in the perception of non-determinism. We therefore believe these patterns deserve close attention.

- The first pattern was found in most of the solutions in the category "almost deterministic". These solutions include automata which are basically deterministic, but contain local non-determinism, expressed in a few non-deterministic transitions. In most cases, due to the deterministic character of the automaton, these transitions are redundant. An example of such an automaton can be seen in Figure 6 which shows an automaton for condition 1 in the language definition. This automaton is deterministic in nature, except for the initial state, in which there are two transitions with the letter b: the first returns to the initial state, and the second is to q_1. The self loop with b in the initial state is redundant, though it does not violate the correctness of the automaton. However, its existence indicates an only partial understanding of the non-deterministic mechanism. Interestingly enough, most of the redundant non-deterministic transitions that we found were self loops in the initial state. Indeed, a self loop transition, with all the alphabet letters, can be found in many of the examples of NFAs in the CM textbook, and some of the students may identify the non-deterministic model with such a transition.

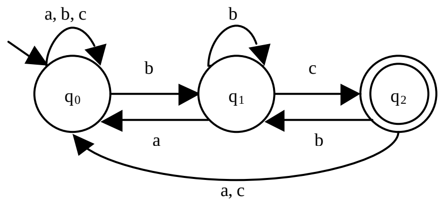

FIGURE 6
LOCAL NON-DETERMINISM IN AN ALMOST DETERMINISTIC AUTOMATON

- The second pattern is "symmetric" to the first. It can be found in most of the solutions in the category “almost non-deterministic”. These solutions include automata which are basically non-deterministic, but contain a few transitions which would be found in the deterministic version of this automaton, but are redundant in the non-deterministic automaton. An example of such an automaton is shown in Figure 7, which is an automaton for condition 1 in the language definition. This automaton is non-deterministic in nature, except for the state q_1, in which there is a redundant self loop transition with the letter b. Again, this transition doesn’t violate the correctness of the automaton. However, its existence indicates only a partial understanding of the non-deterministic mechanism.

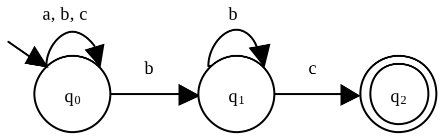

FIGURE 7
REDUNDANT TRANSITIONS IN A NON-DETERMINISTIC AUTOMATON

- In some cases, we could identify traces of the solution process in a student’s answer. Sometimes students wrote a few preliminary versions which they chose not to complete. In the few such cases we encountered in this data, the process indicates a change from a non-deterministic model to a deterministic one. That is, the early versions are non-deterministic or almost non-deterministic, but as the students “improve” the solution they construct almost deterministic or fully deterministic automata. In some of these cases, the preliminary versions were indeed incorrect, but usually only simple and local corrections were necessary. It seems that after recognizing a problem in the automaton, the students preferred to shift to the deterministic, and perhaps more familiar, model, instead of correcting the mistake within the non-deterministic model.

- The fourth and last pattern was found among the fully deterministic solutions, or the solutions in which some or all of the automata for the sublanguages were deterministic. In these cases, the students constructed incorrect NCDFAs (incorrect in the sense that they don’t accept the required language). The students utilized the freedom of omitting necessary transitions, which is characteristic of the non-deterministic model, without introducing the non-deterministic transitions that enable this. An example of such a solution is shown in Figure 8, which presents an incorrect automaton for condition 1. In this automaton, there are no transitions with a and b in q_1, and no transitions in q_2. These transitions cannot be omitted in any correct automaton that accepts that language, unless it contains non-deterministic behavior with b in the initial state. So, even though the automata constructed in this pattern were deterministic in nature, the error stemmed from a partial understanding of the non-deterministic mechanism.

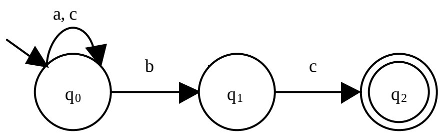

FIGURE 8
INCORRECT NON-COMPLETE DETERMINISTIC AUTOMATON

We emphasize again that a solution that matches one of these patterns may indicate a problem in the perception of non-determinism, even when the solution is correct.

INTERVIEWS

In the beginning of 2003, we conducted four interviews with students who had finished studying the fourth chapter of the CM unit a few weeks before, and had been tested on the material a week before. Through these interviews we hoped to gain some insight into the solution process, and the reasons for choosing one model over the other. Students with various levels of achievement were chosen by the teacher, who did not know in advance what question they would be asked. The four students were asked to solve the question discussed above. After completing their first version of the solution, they were asked a few questions regarding decisions they made when solving the problem. Three of the students gave a non-deterministic solution (two of them performed a direct construction and one decomposed the language into two sublanguages). One of these students was not very cooperative and we were unable to glean any information regarding his decision to use the non-deterministic model. The other two students said that they thought that it was not possible to construct a deterministic automaton for this language. The fourth student used a deterministic approach. He decomposed the language into three sublanguages and constructed one DFA and two NCDFAs. He said he always preferred the deterministic model because it suits him. He described himself as a person with a tendency toward the exact sciences (physics) and in his opinion, non-deterministic thinking is not consistent with that. However, he added that he had no problem with the non-complete deterministic model. This student's explanation indicates a predetermined preference for the deterministic model over the non-deterministic one, irrespective of language. Such a preference could perhaps be changed if the teacher emphasized the non-deterministic model. In this specific case, the teacher reported that she herself felt more comfortable with the deterministic model. The answers given by the two students who chose to use the non-deterministic model show that technical knowledge of the model does not necessarily reflect full understanding, and in particular, a full perception of its computational capabilities.

DISCUSSION AND CONCLUSIONS

Our results show a significant tendency towards the deterministic model. However, since we didn't ask the students specifically to construct a non-deterministic automaton, they had the freedom of choice. It is possible that if the question had been asked differently, the students might have successfully constructed an NFA. However, we believe that the fact that, given the choice, they preferred the deterministic model is in itself an important indicator of their level of understanding of the non-deterministic model. In addition, the few patterns we found in the students answers also indicate an only partial understanding of the non-deterministic model.

No significant differences in the use of determinism was found when the results were analyzed by grade level or level of mathematics. Such differences were indeed found for the correctness of solutions for this and other chapters of the CM unit. This will be discussed elsewhere.

The fact that in some of the classes, the ratio of deterministic to non-deterministic solutions was not the same as that found for the entire research population, suggests that the teaching process can affect the students' tendency. Specifically, if the teacher emphasizes and demonstrates the advantages of the non-deterministic model, the students tend to use it more.

The unexpected answers given by two of the students interviewed, that they didn't think constructing a DFA was possible, indicate a problem in understanding the theory underlying the non-deterministic model. Even if students construct non-deterministic automata freely and correctly, the teacher cannot assume that they fully understand the theoretical meaning of this model. Specifically, students may not realize that the deterministic and non-deterministic models are equivalent. Therefore, the teaching process should emphasize the theoretical aspects and not only the technical aspects. Indeed, studies dealing with the perception of non-determinism that focus on the teaching process, using classroom observations and interviews with teachers, might be helpful.

Our results show that the concept of non-determinism is a difficult one for students to understand. However, since it is one of the basic topics of CM, it is important for students to understand it properly. Full understanding of the non-deterministic model can affect students' comprehension of other topics in CM, such as pushdown automata and context free languages, since the pushdown automata model is also a non-deterministic model. Thus, special effort should be made to prepare teachers for this unit, to ensure that the teaching process in class properly emphasizes the non-deterministic model in its theoretical and technical aspects.

Even though non-determinism is a basic computational and mathematical concept, the CM unit we developed is currently the only part of the high school curriculum which introduces the concept. The perception of non-determinism has never previously been examined in a high school or an academic context. We conducted this research with high school students, but since the issues are relevant to college and university students as well, we plan to conduct such a study in the near future. We also intend to conduct more interviews with high school students, to gain even more insight into the process of choosing the model, while solving questions dealing with finite automata.

REFERENCES

[1] Gal-Ezer, J., Beeri, C., Harel, D. and Yehudai, A. "A High School Program in Computer Science", *Computer,* Vol. 28, No. 10, 1995, pp. 73-80.

[2] Gal-Ezer, J. and Harel, D., "Curriculum and Course Syllabi for a High-School Program in Computer Science", *Computer Science Education* Vol. 9, No. 2, 1999, pp. 114-147.

[3] "Computational Models", A Textbook and a Teacher's Guide, The Open University, Israel, 1998 (In Hebrew).

[4] Merritt, S. M. et al., "ACM Model High School Computer Science Curriculum", *The Report of the Task Force of the Pre-College Committee of the Education Board of the ACM*, 1994, pp. 1-25.

[5] Atchison, W. F. et al., "Curriculum '68, Recommendations for Academic Programs in Computer Science", *Comm. of the ACM*, Vol. 11, No. 3, 1968, pp. 151-197.

[6] Denning, P. J. et al., "Computing as a Discipline", *Comm. of the ACM*, Vol. 32, No. 1, 1989, pp. 9-23.

[7] Tucker, A. B. et al, "Computing curricula 1991, A summary of the ACM/IEEE Joint Curriculum Task Force Report", *Comm. of the ACM*, Vol. 34, No. 6, 1991, pp. 69-84.

[8] IEEE Computer Society/ACM Task Force, "Year 2001 Model Curricula for Computing (CC-2001)", *http://www.computer.org/education/cc2001/report/*, 2002.

[9] Hopcroft, J.E. and Ullman, J. D., "Introduction to Automata Theory, Languages and Computations", Addison-Wesley, Reading, Ma.. 1979.

[10] Gal-Ezer, J. and Zur, E., "The Concept of 'Algorithm Efficiency' in the High School CS curriculum", *FIE 2002, http://fie.engrng.pitt.edu/fie2002/index.htm, 2001.*