CAS Based Approach for Discussing Subset Construction

György Maróti, Szeged (Hungary)

Abstract: This paper continues the discussion of teaching nondeterminism (see [6]) where we presented a didactic approach introducing the notion of nondeterministic automata. Although in this paper we use the same methodology we have to face up to new didactic challenges. Namely, teaching the subset construction requires answers to the question how can CAS be used in teaching the different phases of mathematical problem solving so that we can reach higher cognitive efficiency.

Kurzreferat: Dieser Artikel setzt die Behandlung der Nichtdeterminiertheit aus [6] fort, wo ein didaktischer Zugang zur Einführung nichtdeterministischer Automaten präsentiert wurde. Obwohl die gleiche Methodologie eingesetzt wird, treffen wir auf neue didaktische Herausforderungen. Die Vermittlung der Teilmengen-Konstruktion erfordert nämlich eine Antwort auf die Frage, wie ein CAS eingesetzt werden kann, um die verschiedenen Phasen im mathematischen Problemlösungsprozess so zu gestalten, dass eine höhere kognitive Effizienz erreicht wird.

ZDM-Classification: C20, D40, H70

1 Introduction

Automata constructions are widely used in the theory of automata. When we prove, for example, the closure properties of the class of acceptable languages we take two automata and construct a third one which recognizes the union or the intersection of two languages accepted by the original automata, respectively.

Generally speaking an **automata construction** is a procedure which creates a new automaton from one or more automata with a specific goal. In this general sense the algorithm which construct the minimal state automaton recognizing a certain language can also be considered as automata construction.

In this paper we focus on **subset construction** (see [4]), the tool with which we can create equivalent deterministic automaton. In this sense the existence of subset construction proves one of the basic results of automata theory which states that the recognition power of deterministic and nondeterministic automata coincides with each other.

Our didactic method is that we present a **guided tour** (see [1], [2] and [6]) where we touch every milestones of the mathematical problem solving. In the **motivation phase** we raise the problem, ask questions and point out that the existence of a construction, which create equivalent deterministic automaton is surprising fact. We do not state the exact theoretic result instead we only circumscribe it. This serves first of all for appointing the goal to be reached by the end of the tour.

Next we get acquainted with the usage of tool, a Maple procedure of `aut` package (see [6]), which implements the construction whose existence we have indicated in the motivation phase.

Of course we use this procedure as blackbox. At this phase our aim is to get the device for experiment. We take a nondeterministic automaton and by means of this tools we generate the equivalent deterministic automata. Then we make several observations by comparing the states and the operation of these automata. According to the rule of four (see [5]) our observations are based on the transition graphs (**graphical representation**), while the results of observations are worded first and then making the paraphrase more exact we express the results symbolically (**symbolic representation**).

Although we always choose examples which clearly shows the correlation between our data it is necessary to highlight the importance of managing the special cases. Special cases are often degenerate which may be confusing and render the understanding significantly difficult.

We organize Maple worksheet in such a way that it can facilitate the independent exercise for the students. The **PoP** (Point of Practice) and **EoP** (End of PoP) pairs (see [6]) designate special parts of the worksheet, which can be carried out several times. The number of repetitions can vary student by student. Nevertheless the input data which the subsequent commands depend on can be found at the PoP point where the students find also hints for other proposed values of parameters.

After we have discovered the correlation we leave **observation phase** and enter the **abstraction phase** where our task is to generalize the results of the observations which we have performed on concrete states and transitions of concrete automata. The first level of abstraction is performed when we formulate a statement for every state and every transition of the concrete automata, while in the second level abstraction we generalize the statement for arbitrary automata.

This step is executed technically by introducing new mathematical notion, namely the notion **of subset automata** and the **subset construction**. What we do is to give a name of the phenomenon we discovered by which we create its **mathematical model**.

After the strict discussion of this model we end our guided tour by giving exact **proof** for the theorem we stated at the beginning of our discussion.

2 . The subset construction

2.1 Motivation

Our notion of automaton is nondeterministic (see [3], [4] and [6]). The nondeterministic behavior is a consequence of the feature that an automaton can have more than one initial state and/or the result of transition function is not necessarily uniquely determined by the current state and input signal.

Deterministic automata are special (nondeterministic) automata which implies the trivial consequence that if a language can be accepted by deterministic automaton it can be accepted by (nondeterministic) automaton as well. Less obvious is to answer the inverted question. If a language is acceptable by a nondeterministic automaton whether can it be accepted by a deterministic automaton? In other words, we may ask if the recognition power of nondeterministic automata is stronger or not than that of deterministic automata?

At a first glance we tend to answer yes. It seems that we have several arguments, which strengthen this feeling. For first nondeterministic automata are more complex tools, their stochastic nature may allow much complicated work. On the other hand we feel sure there exist much more nondeterministic automata than deterministic ones and the larger is the class of automata the larger must be the class of languages acceptable by automata belonging to this class.

In spite of these arguments seem realistic we are wrong. One of the basic results of automata theory is that nondeterministic automata own the same level of recognition power as deterministic automata. This statement can be expressed more simply and clearly by means of the notion of equivalence. Two automata are called to be **equivalent** if both of them recognize the same language. What we can state now is for every nondeterministic automaton there exists equivalent deterministic automaton. The proof of this statement is based on an automata construction, which takes an arbitrary (nondeterministic) automaton and creates an equivalent deterministic one.

2.2 The Tool

The construction of equivalent deterministic automaton is implemented in **aut** package by the procedure **ConDet** (**Con**struct **Det**erministic automaton). In this section we learn how to use this procedure. To begin with we activate the package and generate the nondeterministic automaton Ξ. As for using the other procedures of **aut** package please refer to [6].

```
> with(aut):
> Xi:=genaut([a,1,b,b,{0,1},b,b,0,c,
             c,0,f]);
```

$$\Xi := [\,\{a, f, b, c\},\ \{0, 1\},\ \delta,\ \{a\},\ \{f\}\,]$$

```
> printaut(Xi);
```

$$\left[\begin{array}{ccc} STA \setminus INP & 0 & 1 \\ a & - & \{b\} \\ b & \{b, c\} & \{b\} \\ c & \{f\} & - \\ f & - & - \end{array}\right]$$

Set of initial states: , { *a* }

Set of final states: , { *f* }

Next we use **ConDet** and construct automaton Θ. The calling sequence of **ConDet** is simple. Besides giving the automaton for which we want to construct the equivalent deterministic automaton we can also specify the method of construction. The method can be `define` or `generate` which we discuss later.

```
> Theta:=ConDet(Xi,define);
```

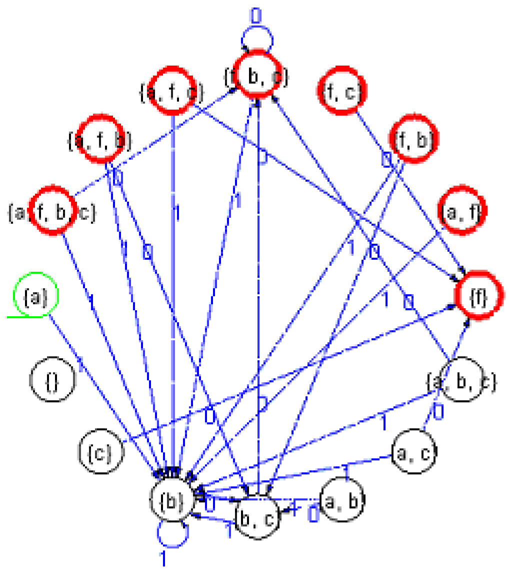
$$\Theta := [\,\{\{\,\}, \{f\}, \{f, c\}, \{f, b\}, \{a, c\}, \{a, b\}, \{a, f, b\}, \{a, f, c\}, \{a, f\}, \{b\}, \{b, c\}, \{c\}, \{a, b, c\}, \{a, f, b, c\}, \{f, b, c\}, \{a\}\},\ \{0, 1\},\ \delta,\ \{\{a\}\},\ \{\{f\}, \{f, c\}, \{f, b\}, \{a, f, b\}, \{a, f, c\}, \{a, f\}, \{a, f, b, c\}, \{f, b, c\}\}\,]$$

```
> type(Theta,deterministic);
```

true

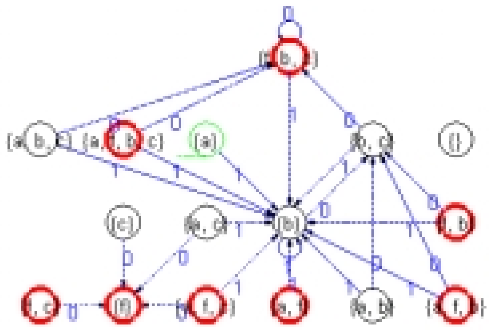
Although Θ is deterministic the output of `ConDet` does not seem to be very encouraging. As a matter of fact it is almost impossible to comprehend it. Things are getting even worth if we plot the transition graph of Θ. We have sixteen nodes and a lot of confused edges.

```
> plotaut(Theta);
```



To demonstrate the visualization power of Maple we display another version of this perplexing transition graph. The **aut** package allows us to determine the position of different nodes which helps us to get a much nicer figure.

```
>p:= {a}=[-1,0],{b}=[0,-1],{ b,c}=[1,0],
      {f,b,c}=[0,1]:
  p:=p,{f,a,b,c}=[-2,0],{f,a}=[0,-2],
      {}=[2,0] ],{a,b}=[1,-2]:
  p:=p,{f,a,b}=[2,-2],{a,b,c}=[-3,0],
      {f,b}=[2,-1],{a,c}=[-1,-1:
  p:=p,{f,a,c}=[-1,-2],{f}=[-2,-2],
      {c}=[-2,-1], {f,c}=[-3,-2]:
  plotaut(Theta,p);
```



Let us take a closer look at this graph. What can we asset about the states of Θ? Recall that automaton Ξ possesses states {a,b,c,f}. Yes, we can observe that all the states on the graph are subsets of {a,b,c,f} the set of states of automaton Ξ.

Next let us focus to bold circles on the graph, the final states of automaton Θ. They have one common property. Every final state contains the state f, the final state of automaton Ξ. Make a mental note of these properties now. They will turn out to be useful later on.

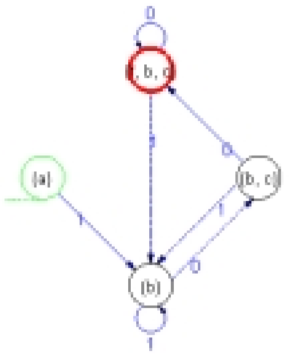
The most important property of this graph,however, is that it helps us to realize that Θ possesses several redundant states. For example the state {} stands alone there is no transition defined on it. Furthermore, there are states, e.g. {c} or {a, c} for which there is no arrow which points at them. States with this property can not be reached from the initial state, which means these states are irrelevant from the point of word recognition.

So that we can eliminate the redundant states **ConDet** offers a second method called `generate`. We use this method to construct another equivalent deterministic automaton.

```
> Phi:=ConDet(Xi,generate);
```

$$\Phi := [\{\{b\},\{b,c\},\{f,b,c\},\{a\}\},\{0,1\},\delta,\{\{a\}\},\{\{f,b,c\}\}]$$

```
> plotaut(Phi);
```



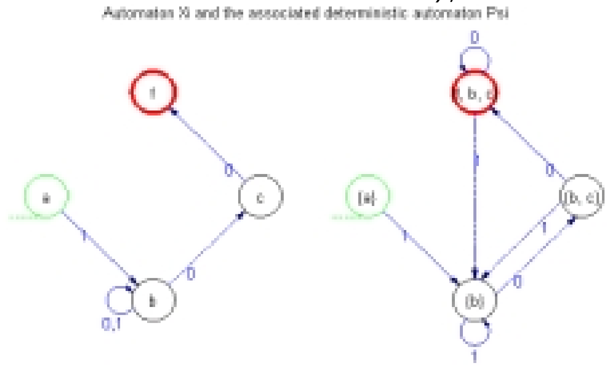
This result is much better. Φ has four states only instead of sixteen and it is **connected**, i.e. every state can be reached from the initial state.

It is worth to spend some time to compare the transition graph of Θ and Φ. Notice the transition graph of Φ is a subgraph of the transition graph of Θ. Recall that both Φ and Θ were created from nondeterministic automaton Ξ by **ConDet** which constructs equivalent deterministic automaton. So both Φ and Θ are deterministic and equivalent with the nondeterministic automaton Ξ. As we see the equivalent deterministic automaton is not uniquely determined.

2.3 Observations

From now on we focus to the comparison of the original nondeterministic automaton Ξ and the constructed deterministic automata. Although we work with Φ our observations always can be checked on automaton Θ. To make the comparison easier we plot the transition graphs of Ξ and Φ side by side.

```
> p1:=plotaut(Xi,[-2,0]):
  p2:=plotaut(Theta,[1,0]):
  plots[display]({p1,p2},
scaling=unconstrained,
title="Automaton Xi and the associated
deterministic automaton Theta");
```



First we examine the relationship of states and transition function. As we have already seen before the states of automaton Φ are subsets of the set of states of automaton Ξ.

To discover the relationship between the transition functions let us start experimenting. In the following command we specify and input signal (xi:=`0`), take the set {b,c}, which is a state of Φ, and calculate the result of transitions $\delta(\Phi.\{b,c\},\xi)$, $\delta(\Xi.,\{b\},\xi)$ and $\delta(\Xi.,\{c\},\xi)$.

```
> xi:=`0`:                          # PoP-EoP
                                    # xi:=`1`
'delta(Phi,{b,c},xi)'=Delta(Theta,{b,c},xi)
;
'delta(Xi,b,xi)'=Delta(Xi,b,xi);
'delta(Xi,c,xi)'=Delta(Xi,c,xi);
```

$$\delta(\Phi,\{b,c\},\xi)=\{f,b,c\}$$

$$\delta(\Xi,b,\xi)=\{b,c\}$$

$$\delta(\Xi,c,\xi)=\{f\}$$

It is easy to see that the set {f,b,c} is the union of sets {b,c} and {f}. We can say that in case of state {b,c} of deterministic automaton the transitions can be performed in such a way that we calculate the transitions in the nondeterministic automaton for the states b and c, respectively, in the end we form the union of the resulting sets.

We formulate symbolically our assertion by the equality

$$\delta(\Theta,\{b,c\},\xi)=\Delta(\Xi,b,\xi)\cup\Delta(\Xi,c,\xi).$$

Notice that the right hand side of this equality is nothing else than $\Delta(\Xi,\{b,c\},\xi)$. So we obtain

$$\delta(\Theta,\{b,c\},\xi)=\Delta(\Xi,\{b,c\},\xi).$$

This equality expresses the fact that in case of the state {b,c}of the deterministic automaton the result of every transition is equal to the result of generalized transition function of nondeterministic automaton.

Let us check our observation for another state of automaton Φ, namely we consider now the one element set {b}.

```
> xi:=`0`:                          # PoP-EoP
                                    # xi:=`1`
'delta(Theta,{b},xi)'=Delta(Theta,{b},xi);
'delta(Xi,b,xi)'=Delta(Xi,b,xi);
```

$$\delta(\Theta,\{b\},\xi)=\{b,c\}$$

$$\delta(\Xi, b, \xi) = \{ b, c \}$$

We obtain that for the one element set {b} automata Ξ Ξ and Φ work in the same way, i.e.

$$\delta(\Theta,\{b\},\xi) = \delta(\Xi,\{b\},\xi)$$

At a first glance we may feel that this result contradicts to our former observation. In case of the set {b,c} the two automata worked differently.

How can we resolve this contradiction? Just note that by definition

$$\delta(\Theta,\{b\},\xi) = \Delta(\Xi,\{b\},\xi),$$

which yields

$$\delta(\Theta,\{b\},\xi) = \Delta(\Xi,\{b\},\xi).$$

This equality is already in full harmony with our first observation.

2.4 Abstraction

Up till now we have seen that for the states {b,c} and {b}of deterministic automaton the result of every transitions coincides with the result of generalized transition functions of the original (nondeterministic) automaton.

It is high time at this point to formulate a more general statement. We feel sure that our observation should remain true for all states of deterministic automaton. More specifically we assert that for every state A of automaton Φ and for every input signal ξ the equality

$$\delta(\Phi, A,\xi) = \Delta(\Xi, A,\xi)$$

holds.

As our statement is about two concrete automata we can calculate all possible transitions of automaton Φ and check the validity of our statement.

```
> xi:=`0`:                              # PoP-EoP
                                        # xi:=`1`:
for s in Phi[1] do
 printf("Phi: %a,%a- %a\n",s,xi,
 Delta(Theta,s,xi),xi);
 for ss in s do
  printf("  Xi: %a,%a->%a\n",ss,xi,
  Delta(Xi,ss,xi))
 od:
od:

Phi: {b},`0`->{b, c}
  Xi: b,`0`->{b, c}
Phi: {b, c},`0`->{f, b, c}
  Xi: b,`0`->{b, c}
  Xi: c,`0`->{f}
Phi: {f, b, c},`0`->{f, b, c}
  Xi: f,`0`->{}
  Xi: b,`0`->{b, c}
  Xi: c,`0`->{f}
Phi: {a},`0`->{}
  Xi: a,`0`->{}
```

The output is self-explanatory. The result of every transition on Φ equals to the union of the corresponding transitions in Ξ

Notice we reached the first level of abstraction. Although our statement is about two concrete automata the statement asserts something for every state and for every input signal of these two special automata. We are on the same time interested in the second level abstraction where we try to generalize our statement for every pair of automaton Ω and Ω_1 (=ConDet(Ω)).

2.5 Concept Formation

At this point of our discussion we reached the most exciting point of mathematical investigations. After we have observed and described the relationship of two concrete automata, instead of trying to proof our statement in general we introduce two new notions. Of course these notions reflects the properties we observed before.

Let $\Omega = [A, X, \delta, Q, F]$ be arbitrary automaton. The automaton $\Omega_1 = [A_1, X, \delta_1, Q_1, F_1]$ is called the **subset automaton** of Ω if

1. $S_1 \subseteq 2^S$ i.e. the set of states of Ω_1 is a subset of the powerset of states of Ω and
2. For every state $A \in S_1$ and signal $\xi \in X$ the equality $\delta(\Omega_1, A, \xi) = \Delta(\Omega, A, \xi)$ holds.

Furthermore if

3. $Q_1 = Q$ and
4. $F_1 = \{A \mid A \in S_1 \wedge A \cap F \neq \emptyset\}$

then we say automaton Ω_1 is constructed from automaton Ω by using the **subset construction**.

It is worth highlighting the essential features of subset construction. Requirements 1. and 2. prescribes algebraic behavior of subset automaton. The states of subset automaton are sets consisting of states of the original automaton. In this way if $A \in S_1$ then $A \subseteq S$. The reverse statement is not necessarily true. A subset of states of Ω may not be a state of subset automaton.

As the result of generalized transition function $\Delta(\Omega, A, \xi)$ is a uniquely determined subset of S we obtain that $\delta(\Omega_1, A, \xi)$ is a uniquely determined element of S_1. This yields that Ω_1 is deterministic.

An automaton may have more than one subset automaton. Namely choosing $S_1 = 2^S$ we obtain the largest subset automaton in the sense that its set of states includes the set of states of every subset automaton. Of course we are much more interested in the smallest subset automaton which fulfills requirements 3. and 4.

In the end we emphasize the fact that an automaton and its subset automaton work with the same set of input signal. This is absolutely necessary so that we can speak of whether they recognize the same language.

2.6 Mathematical Discussion

Take an arbitrary state A of Ω_1, an input word w over the common alphabet and try to imagine that automata Ω and Ω_1 operate concurrently.

As for automaton Ω, it reads the first input signal ξ and because A is a subset of its states we have to use the generalized transition function to determine the new subset $A_1 = \Delta(\Omega, A, \xi)$.

The subset automaton is deterministic and A is its state. It reads the first input signal and moves to the state

$$\delta(\Omega_1, A, \xi) = \Delta(\Omega, A, \xi) = A_1$$

according to its definition. So the new state of Ω_1 is nothing else than set A_1 .

Notice, before reading the second input signal both automata are in similar position as before reading the first signal. In this way when reading the second signal similar things should happen. The automaton Ω determines a new subset A_2 whilst the deterministic automaton Ω_1 moves to the state A_2 ,etc.

We can observe this phenomenon by looking at the output of the following command.

```
> R:={b,c}:                     # PoP-EoP
                                # R:={a}
                                # w:=`100`
w:=`011`:
Delta(Xi,R,w,tr2);
Delta(Phi,R,w,tr2);

-Delta begins with {b, c}
 .b,`0`->{b, c}
 .c,`0`->{f}
-Delta({b, c},`0`) = {f, b, c}
 .b,`1`->{b}
-Delta({f, b, c},`1`) = {b}
 .b,`1`->{b}
-Delta({b},`1`) = {b}
```

$$\Delta(\{b, c\}, 011) = \{b\}$$

$$\{b\}$$

```
-Delta begins with {b, c}
 .{b, c},`0`->{f, b, c}
-Delta({b, c},`0`) = {f, b, c}
 .{f, b, c},`1`->{b}
-Delta({f, b, c},`1`) = {b}
 .{b},`1`->{b}
-Delta({b},`1`) = {b}
```

$$\Delta(\{b, c\}, 011) = \{b\}$$

$$\{b\}$$

This output clearly shows the nondeterministic operation of automaton Ξ and the deterministic nature of Φ, while we can see that every step of calculation results in the same subset of states.

What we have observed is that the generalized transition function of a (nondeterministic) automaton works in the same way as the generalized transition function of its subset automaton restricted to the one element sets of states.

2.7 Theorem and Proof

We state this assertion more exactly and more formally in the next

Remark

If Ω_1 is a subset automaton of automaton Ω then for every state A of Ω_1 and for every input word w

$$\Delta(\Omega_1, A, w) = \Delta(\Omega, A, w)$$

Proof

The proof is performed by induction on the length of the word w.

The statement is obvious for the empty word as both $\Delta(\Omega_1, A, \varepsilon)$ and $\Delta(\Omega, A, \varepsilon)$ equal to A.

Let us now assume the statement is valid for the word w and consider the input word $w\xi$. We obtain

$$\begin{aligned} &\Delta(\Omega_1, A, w\xi) = \\ &= \delta(\Omega_1, \Delta(\Omega_1, A, w), \xi) = \\ &= \delta(\Omega_1, \Delta(\Omega, A, w), \xi) = \\ &= \Delta(\Omega, \Delta(\Omega, A, w), \xi) \\ &= \Delta(\Omega, A, w\xi) \end{aligned}$$

This ends the proof of our Remark.

Theorem

Every automaton is equivalent with the automaton constructed from it by using the subset construction.

Proof

Consider an automaton $\Omega = [A, X, \delta, Q, F]$ and let us assume that $\Omega_1 = [A_1, X, \delta_1, Q_1, F_1]$ is constructed from Ω by using the subset construction.

By definition Ω recognizes the input word w if and only if

$$\Delta(\Omega, Q, w) \cap F \neq \varnothing .$$

As $Q = Q_1$ using the remark above we obtain

$$\Delta(\Omega, Q, w) = \Delta(\Omega_1, Q, w) = \Delta(\Omega_1, Q_1, w)$$

In this way

$$\Delta(\Omega, Q, w) \cap F \neq \varnothing$$

if and only if

$$\Delta(\Omega_1, Q_1, w) \cap F \neq \varnothing .$$

Because the later condition is equivalent with the statement that

$\Delta(\Omega_1, Q_1, w)$ is final state of Ω_1

we have obtained that Ω recognizes an input word if and only if Ω_1 also recognizes the same word. As this means that Ω and Ω_1 accept the same language, we have proved their equivalence.

This is the theorem we have yearned. It provides us with the ability that for every nondeterministic automaton we can create equivalent deterministic automaton, which proves that the recognition power of nondeterministic automata is the same as that of deterministic automata.

3. Conclusion

We have roved over a long way. We were eager to compare the recognition power of deterministic and nondeterministic automata. First we learned the tool with which we were able to make experiments. We observed the basic features of concrete automata, generalized our observations and took part in the most exciting adventure of mathematics, in the introduction of new notions.

Having the knowledge we collected during our journey in the end we easily proved the theorem, which gave us the answer for our original question.

We truly hope this article presents a newer example for guided tour making evidence that computer algebra systems like Maple are the adequate tool for doing motivated and enjoyable mathematics as well as in the field of automata theory.

References

[1] A. Ambrus, Matematikadidaktika, (1995)
[2] K.J. Fuchs, Computer Algebra Systems in Mathematic Education, International Symposium - Anniversary of Pollack Mihály College of Engineering, (2002)
[3] J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison- Wesley, Reading, MA, (1979)
[4] Kozen: Automata and Computability. - Springer-Verlag, New York, (1997)
[5] W. Lindner, CAS-Supported Multiple Representations in the Elementary Linear Algebra, The case of Gaussian Algorithm, International Symposium - Anniversary of Pollack Mihály College of Engineering, (2002)
[6] Gy. Maróti, Didactic Approach for Teaching Nondeterminism in Automata Theory,
ZDM, 2003 Vol. 35 (2)
[7] M.B. Monagan, K.O Geddes, K.M. Heal, G.Labahm, S.M.Vorkoetter, J. McCaron. P.DeMarco: Maple 8 Advanced Programming Guide -Waterloo Maple , (2002)
[8] M.Y. Vardi, An Automata-Theoretic Approach to Linear Temporal Logic