Animation of Relations between Computational Models and Their Language Representations

Alexsandro F. da Fonseca*[1], Alison H. R. da Silva**[1], Mauricio Ayala-Rincón[2], Haydée Werneck Poubel[2], and José de Siqueira[3]

[1] Departamento de Ciência da Computação, Universidade de Brasília, Brasil
alex@cic.unb.br alison@cic.unb.br
[2] Departamento de Matemática, Universidade de Brasília, Brasil
ayala@mat.unb.br haydee@mat.unb.br
[3] Departamento de Ciência da Computação, U. Federal de Minas Gerais, Brasil
jose@dcc.ufmg.br

Abstract. We discuss how to increase and simplify the understanding of the equivalence relations between machine models and language representations of formal languages by means of the animation tool *SAGEMoLiC*, currently in development. The philosophy of our new educational tool goes further than these of the usual ones, since it permits the simulation of the execution of models of computation, as many other animation systems do, as well as the animation of the proofs of the corresponding equivalences. In contrast with previous systems, our philosophy to animate equivalence theorems is not a simple visualization of the direct translation between computational models and their grammatical representations, because we make emphasis in the key theoretical notions involved in the formal proofs of these translations.

Keywords: *Automata theory, Formal languages, Visualization, Algorithm Animation*

1 Introduction

Most of the basic notions studied in formal language and automata theory can be illustrated and visualized by means of graphical representations as state or transition diagrams and grammatical derivation trees. Some of these concepts can be easily understood by interacting with computer simulations of the involved machine models. For instance, the mechanics of finite state automata and Turing machines can be quickly assimilated after some simulations of simple examples of these machines with systems as *Deus ex Machina* [Tay98], *JFLAP* [HR00], *Turing Word* [BE95] etc. Simulation of specific machines is a very productive exercise in order to understand each kind of machine, but the most important concepts of formal language and automata theory are the ones that establish the relations between these models, their grammars and languages.

* Supported by CAPES Brazilian Council.
** Partly supported by CNPq Brazilian Foundation.

These relations are established by constructive proofs that allow, from a specific instance of a model or language representation as input, the construction of the related model or language representation as output. For example, a non-deterministic finite state automaton can be transformed into a deterministic finite state automaton; a context-free grammar into a push-down automaton; etc. Implementation of one of the related translations or conversions is an easy programming exercise, that surely will make clear the subjacent equivalence relation (at least for the programmer him/herself). But experimentation with the direct transformations generated by these kind of implementations is not the best way to make clear to students these relations, since in many cases very important factors as the *correctness* of the related translation algorithms are not evident.

From the previous perspective, we explain here how our visualization system *SAGEMoLiC*[1] can be enriched in order to make possible a real explanation of relations of equivalence between computational machine models and their corresponding language representations. *SAGEMoLiC*, that is a Java applet loadable from any web browser, in its current state, enables the construction of finite state automata and the subsequent simulation of their execution with different inputs. Additionally, this system allows for direct transformations between the different representations of the class of languages associated with these machines, that are the regular languages and different kinds of finite state automata as presented in most of the textbooks on automata and formal language theory. Extensions of the philosophy of visualization of equivalence relations here presented are of course possible for different classes of formal languages and their related machine models and grammatical representations.

2 Background: conversions between machine models and their language representations

We assume familiarity with the notions of formal language and automata theory as presented in the textbooks [HU79,LP81,FB94,Kel95,Koz97,Sud97,Tay98]. All our argumentation will be presented based on the tool *SAGEMoLiC* that we will briefly introduce here.

In its current state *SAGEMoLiC* is a full active *visualization tool* or animator of finite state and push-down automata that allows for direct transformations or conversions between the involved machine models and their language representations. For *active* we mean that the user has to provide his/her own machine models and/or language representations for being further executed and simulated according to appropriate inputs given also for him/her. Consequently, *SAGEMoLiC* is adequate for being used for a person with good knowledge of the field, who is able to select adequate examples to explain specific situations

[1] Available at `www.mat.unb.br/~ayala/TCgroup/SAGEMoLiC/`. *SAGEMoLiC* is a Portuguese acronym for *Sistema de Animação Gráfica de Teoremas de Equivalência entre Modelos e Linguagens Computacionais.*

and notions related with the theory. The tool was made as *flexible* and *dynamic* as possible: for instance, during a simulation of execution of a finite state automaton, the user can interrupt the simulation, return to the beginning of the execution, change the input, speed up or slow down the simulation, etc. This flexibility makes very easy the use and adaptation of the system for the explanation of dependent notions after some basic concepts are understood via slow simulations or the use of very simple examples. This makes the tool adequate for explanation of a sequence of concepts of increasing complexity, that is typical from the area. Finally, other of the nice properties of *SAGEMoLiC* is its *portability*: being it a Java applet, the system can be used independently of computer architecture and operational system by means of any web browser on the net.

Most of the currently known systems, as for instance *Deus ex Machina* [Tay98] and *Turing's Word* [BE95], for machine model and related language visualization are simple simulators and are not concerned with the visualization of relations between machine models and their languages. From this point of view Rodger's system *JFLAP* (see [BLP+97]) is surely more interesting, since it is not only a simulator, but it also allows for some *direct conversions* or *translations* between machine and grammatical language representations. Moreover, in the last papers describing the current evolution of the system it is purposed to *granulate* the steps in the transformations to make clearer, at least, these transformations [GR99,HR00].

For the case of the finite state automata machine model and the regular languages we have the possibilities of direct conversions or transformations presented in Figure 1, where as usual the acronyms λ-NDFA, NDFA, DFA, μ-DFA, RE and RG stand for Non-Deterministic Finite Automata with λ-transitions, Non-Deterministic Finite Automata, Deterministic Finite Automata, Minimal Deterministic Finite Automata, Regular Expressions and Regular Grammars, respectively.

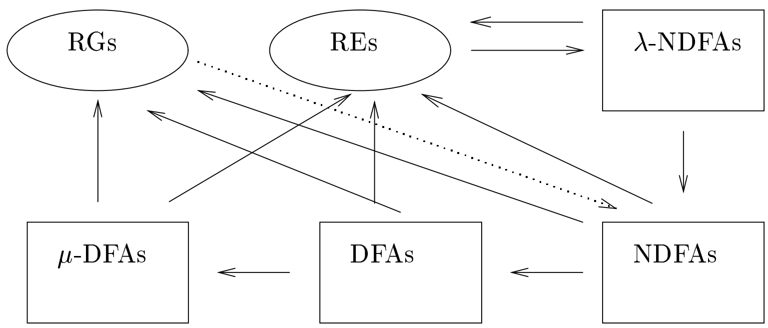

Fig. 1. Possibilities of direct conversions with *SAGEMoLiC*

Once these transformations have been implemented a good tentative to illustrate the relations between the involved models or language representations is to granulate, as proposed by Rodger, for different examples, the steps of the conversion. This works well in the very simple cases such as construction of a a RG from a DFA or construction of a λ-NDFA from a RE, where the equivalence relation is straightforward. But doing that for the more complex and interesting cases doesn't work at all. In these cases, because of the complexity of the transformation, students (and teachers too) have the tendency to focus on the correct execution of the algorithm (and their particular data structure) missing the most important that is the equivalence relation between the models and language representations involved. Observe that this kind of exercise can be done by use of sophisticated debuggers or program animators, but this should not be the objective of a system for illustrating the equivalence relations as the ones we want.

In the remaining of this section, to make the previous discussion clear, we will explain the problems that come arise when trying to explain in this way perhaps the most complex equivalence relation in the context of regular languages: for any DFA there exist a unique equivalent minimal DFA. We will use the standard notation of DFA, as in [HU79], where they are represented by quintuples of the form $\langle Q, \Sigma, \delta, q_0, F\rangle$, being Q, Σ, δ, q_0 and F, respectively, the set of states, the alphabet, the transition function, the initial state and the set of accepting or final states.

The minimization method consists on the determination of *indistinguishable* states by examining whether the result of processing any word beginning from two different states gives rise to identical answers; that means, whether for any word following the transition function beginning from these states one finishes always in the same class of states: final or non-final. As it is well-known, the classical DFA minimization algorithm, makes this by, firstly, ordering the set of states; afterward, assuming pairs of final and non-final states *distinguishable* and then by systematically comparing pairs of states, p, q, in a table according to the image states of the transition function for each symbol of the alphabet: $\delta(q, a)$ and $\delta(p, a), \forall a \in \Sigma$. In the case that the image states are *distinguishable* one can infer the *distinguishability* of states q and p. In the other case, the *distinguishability* of states q and p depends on the eventual *distinguishability* of the image states. We will remind this translation algorithm with a very simple DFA illustrated in Figure 2.

After *distinguishing* between final and non-final states we obtain the table in Figure 3. Subsequently, one has to analyse non-marked pairs of states $(q_0, q_1); \ldots; (q_0, q_6); (q_1, q_2); \ldots; (q_1, q_6); \ldots; (q_4, q_5)$, according to their transitions (for each symbol of the alphabet). For example, states (q_0, q_1) can be marked as *distinguishable* since $\delta(q_0, 0) = q_2$ and $\delta(q_1, 0) = q_5$, being q_2 and q_5 previously marked as *distinguishable*. Also, the *distinguishability* of the pair (q_1, q_3) depends on the ones eventually computed of either (q_4, q_5) or (q_3, q_6), since $\delta(q_1, 0) = q_5$ and $\delta(q_3, 0) = q_4$ and $\delta(q_1, 1) = q_3$ and $\delta(q_3, 1) = q_6$. After analyzing each non-marked pair of states in the table, the non-marked pairs are

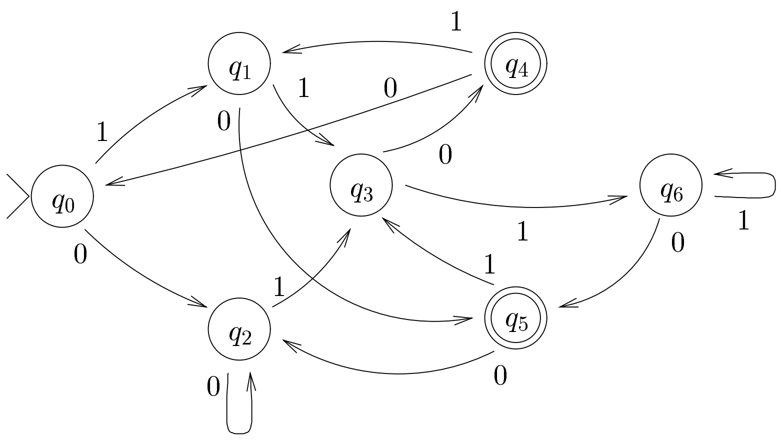

Fig. 2. A simple non-minimal DFA

concluded *indistinguishable* and merged as a single state. The fully completed table and the corresponding automaton (that the reader could recognize as the ones for computing binary numbers congruent to 2 modulo 4) are presented in the Figure 4.

Well, now the question if whether illustrating to students the execution of this translation algorithm (or another specific one) is useful for explaining the existence of a μDFA equivalent to each DFA? And our answer is no, because visualization of a specific conversion algorithm explains details about the algorithm itself and not about the equivalence relation between computational models. Students are usually concentrated on the data structures involved and perhaps on the terminality of the method, but not about what is really making the algorithm on the states. After some visualizations of this conversion for different DFAs they cannot answer very important questions about the models of interest such as:

1. Why closing the *distinguishability* relation between states in this way produces as result sets of *indistinguishable* states?
2. Why set of *indistinguishable* states conform equivalence classes?
3. Why is the resulting DFA deterministic?
4. Why is the resulting automaton a minimal DFA?
5. Why is the resulting automaton unique?

3 A real visualization of equivalence relations between machines and their language representations

Since our intention is to visualize the equivalence relations and not simply to granulate specific translations, we have to analyse case by case each equivalence relation between machine models and language representation and decide

	q_0	q_1	q_2	q_3	q_4	q_5
q_1						
q_2						
q_3						
q_4	$\times$	$\times$	$\times$	$\times$		
q_5	$\times$	$\times$	$\times$	$\times$		
q_6					$\times$	$\times$

Fig. 3. DFA Minimization

	q_0	q_1	q_2	q_3	q_4	q_5
q_1	$\times$					
q_2		$\times$				
q_3	$\times$		$\times$			
q_4	$\times$	$\times$	$\times$	$\times$		
q_5	$\times$	$\times$	$\times$	$\times$		
q_6	$\times$		$\times$		$\times$	$\times$

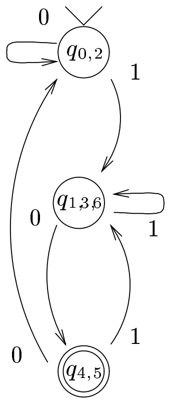

Fig. 4. Final table and corresponding μDFA

whether it is possible to illustrate that via simulation of an specific translation algorithm. In the more complex cases visualizing a specific translation is not enough for explaining the related equivalence relation.

We continue our example of the previous section. Our intention is to illustrate the relation between DFAs and their corresponding minimized DFAs. We think that in this case the most important is to explain and illustrate the relations of *distinguishability* and *indistinguishability* between states and their properties. From a good understanding about that, the students will be able to produce their own conversion or minimization algorithms. More formally, for any DFA $\langle Q, \Sigma, \delta, q_0, F\rangle$, the *indistinguishability* relation to be explained, subsequently denoted by *IDis*, is defined by

$$q\mathit{IDis}\ p \Leftrightarrow (\forall x \in \Sigma^*,\ \text{either } \delta(q,x), \delta(p,x) \in F \text{ or } \delta(q,x), \delta(p,x) \in Q \setminus F)$$

Obviously, the *distinguishability* relation, for brevity denoted by *Dis*, corresponds to the complement of this relation, and a pair p, q of final and non-final states are distinguished by the empty word, since $\delta(p,\lambda) = p \in F$ and $\delta(q,\lambda) = q \in Q \setminus F$. Of course, each state is indistinguishable from itself. Beginning from this point, what should be explained to the students is simply that once the *Dis* relation is built, one obtains automatically its complement, that is the *IDis* relation. Consequently, what should be clarified and understood is that the *Dis* relation can be built from the pairs of final and non-final states. How to complete the *Dis* relation can be directly illustrated on the state diagram of a DFA as represented in *SAGEMoLiC*:

1. To emphasize the beginning point, we paint in two different colors the final and non-final states. All pairs of final and non final states are included in the *Dis* relation.
2. The user chooses an unconsidered pair of states, $\langle p, q\rangle$, that has not been included in the *Dis* relation. These states are highlighted.
3. For each alphabet symbol $a \in \Sigma$ we highlight the corresponding transitions leaving these states and the destiny states: $\delta(p,a)$ and $\delta(q,a)$. For symbols whose destiny states coincide nothing is done. In the other case, if the pair of states $\langle\delta(p,a), \delta(q,a)\rangle$ has not been considered, this step is recursively repeated for the new pair. This is done until either a pair of destiny states that belongs to the *Dis* relation is reached or all possible pairs of destiny states were previously considered.
4. If for some alphabet symbol the corresponding highlighted pair of destiny states has previously been included in the *Dis* relation, then following the highlighted transitions in the reverse direction, all considered pairs of states are included in the *Dis* relation and highlighted transitions and states are played down until no more states are highlighted.
   If all the pairs of destiny states coincide or all possible pairs of destiny states were previously considered, then a message appears saying that each two states conforming the considered pairs are indistinguishable and an animation is done showing their merging and a redrawing of the transitions that leave from and arrive to them.

5. The user can choose another pair of unconsidered states and execute step (2) to (4) until no more states can be included in the *Dis* and *IDis* relations.

We illustrate our method of explanation by showing how this works for the example in Figure 2. Initially, all pairs of final and non-final states are included in the *Dis* relation. Suppose that the user choses the unconsidered pair of states $\langle q_0, q_1\rangle$. From this pair, following the transitions for the symbol 0, we reach the pair of states $\langle q_2, q_5\rangle$ that belong to *Dis*. Then $\langle q_0, q_1\rangle$ is included in the *Dis* relation. Now, suppose that the user selects the unconsidered pair $\langle q_0, q_2\rangle$. The ordering of the analysis is illustrated in Figure 5. Following the two transitions

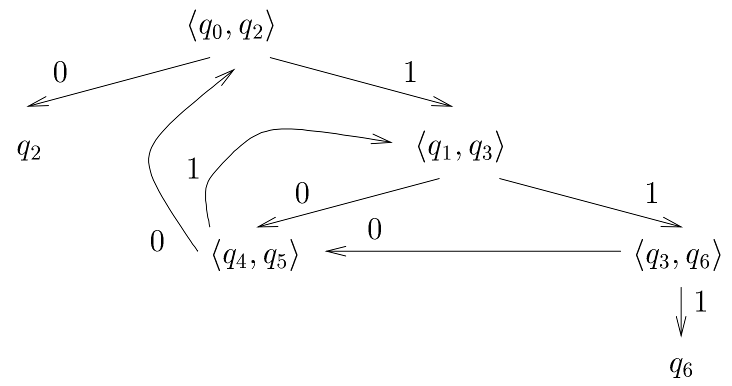

Fig. 5. Ordering of Consideration of Pairs

for 0, $\delta(q_0, 0)$ and $\delta(q_2, 0)$, we reach the sole state q_2 that is indistinguishable from itself. The transitions for 1 give the unconsidered pair $\langle q_1, q_3\rangle$; $\delta(q_0, 1) = q_1$ and $\delta(q_2, 1) = q_3$. Since the pair $\langle q_1, q_3\rangle$ hasn't been considered and doesn't belong to *Dis*, we recursively check its distinguishability. From $\langle q_1, q_3\rangle$ we reach the unconsidered pairs $\langle q_5, q_4\rangle$ and $\langle q_3, q_6\rangle$; $\delta(q_1, 0) = q_5$, $\delta(q_3, 0) = q_4$, $\delta(q_1, 1) = q_3$ and $\delta(q_3, 1) = q_6$. On the one side, the distinguishability of $\langle q_5, q_4\rangle$ depends on the ones of $\langle q_2, q_0\rangle$ and $\langle q_3, q_1\rangle$ that were previously considered in this step. On the other side, the distinguishability of $\langle q_3, q_6\rangle$ depends only on the one of the considered pair $\langle q_4, q_5\rangle$, since $\delta(q_3, 0) = q_4$, $\delta(q_6, 0) = q_5$ and $\delta(q_3, 1) = \delta(q_6, 1) = q_6$. No additional pairs of unconsidered states can be reached neither from the selected pair $\langle q_0, q_2\rangle$ nor from the other pairs involved in the analysis (e.g. $\langle q_1, q_3\rangle$, $\langle q_5, q_4\rangle$ and $\langle q_3, q_6\rangle$). Therefore, the distinguishability of the two states q_0 and q_2 as well as of every two states conforming the considered pairs depend on themselves. Consequently, one can conclude that the states q_0 and q_2 as well as q_4 and q_5 and q_1, q_3 and q_6 are indistinguishable. Joining these sets of states and aggregating the respective transitions we obtain the reduced automaton in Figure 4.

We think that explaining the equivalence relation in this general way allows students (to try) to answer in a coherent manner some of the essential questions about the existence and uniqueness of reduced DFAs that were formulated at the end of the previous section. Moreover, after this kind of visualizations of these relations, that are independent of any (granulation of a) specific translation algorithm, some of the students could give their own algorithms for constructing the *Dis* relation. For instance, they can straightforwardly formulate algorithms that beginning from pairs of non-final and final states and following the transition function in the reverse direction (that is the inverse relation of the transition function) distinguish new pairs of states and so on until no more pairs of states can be distinguished. They can also find some problems such as the lack of a mechanism for eliminating unnecessary states, that is states that cannot be reached from the start ones of from where no final states can be reached.

In its current state, *SAGEMoLiC* allows for the animation previously explained and for a nice visualization of the equivalence relation between (λ-N)DFAs and REs. Additionally, our tool is being improved with adequate animations for true visualizations of the relations between NDFAs and DFAs and λ-NDFAs and NDFAs, that are the relations that we don't consider as obvious to be explained by a simple granulation or debugging of specific translation algorithms between these machine models and language representation. Briefly, our proposal is to explain each of these relations as follows.

– (λ-N)DFAs and REs. On the one side, that the regular language associated with a given RE corresponds to a λ-NDFA is an obvious fact easy to explain. On the other side, our idea to visualize this relation is to show graphically and directly over the state diagram of a (λ-N)DFA how intermediate states (i.e., states that are neither initial nor final) can be eliminated one by one by replacing each combination of "input transition" and "output transition" to and from an specific state by a new transition whose label is the RE built by concatenating
  - the RE labeling the "input transition",
  - the Kleene closure of the union of the REs labeling all the *loop* transitions at this state and
  - the RE labeling the "output transition".

  This is done for each final state, considering the remaining accepting states as non final ones, until all intermediate states are deleted and only the initial and the current final state remain. Then one obtains easily a RE for all words accepted by this final state. This graphic visualization is done directly over the state diagram and it illustrates the principle in that all algorithmic approaches for this transformation are based. In particular, this visualization is independent of any indexing of states and alphabet symbols and subsequently inductive or recursive construction of corresponding REs as done in [HU79,FB94,Koz97,Sud97,Tay98]. From the understanding of this principle students will be able to built his/her own translation algorithmic methods to construct a RE corresponding to a given (λ-N)DFAs. Moreover they will be able to provide improvements on the input (λ-N)DFAs as for example elimination of unnecessary states replacement of the set of final states for a sole accepting state, etc.

- NDFAs and DFAs. That non determinism is an inessential property of finite automata can be graphically visualized over the state diagram of a given NDFA by illustrating the principle that enables the replacement of states by sets of states obtaining an equivalent DFA. Beginning from the given input NDFA, that is preserved during the whole process in a canvas, the user can select a (set of) state(s) and for each alphabet symbol, say a, the corresponding output transitions and the set of destiny states are highlighted. Simultaneously, in another canvas the automata in construction is incremented by including when necessary a sole origin state for the selected (set of) state(s) and another one for the set of destiny states connected by a transition labeled by a. In this way students can be taught about the key notion used in all translation algorithms for building NDFAs from DFAs. After understanding that they will be able to think about efficient and improved methods for this conversion.
- λ-NDFAs and NDFAs. The principle for eliminating empty transitions in λ-NDFAs can be graphically visualized over the state diagram of the automata independently of any translation algorithm. The user selects a state and the system will highlight the selected state and all the states reachable via a sequence of empty transitions from this state. Then, for each alphabet symbol, say a, the states reached via transitions labeled by a from the highlighted states are highlighted by using a different color. In another canvas, where a copy of the states of the given λ-NDFAs has been initially drawn, transitions labeled by a from the state corresponding to the selected one to the states corresponding to the formerly highlighted states are built. Again, the user can built a NDFA equivalent to the given λ-NDFA by means of his/her selection of states and independently of any indexing of states and alphabet symbols and of any algorithmic method.

By illustrating (in this way) the key principles subjacent to the previously equivalence relations, we think a solid explanation will be offered to the students. From this point, surely an interesting and enriching exchange about properties and relations between these machine and grammatical language representation will take place. Of course, being this visualization independent of any translation method, an important part of this exchange will be focused on how to implement efficient translation algorithms.

We consider all other relations between machine models and language representations of regular languages very obvious to deserve special visualizations. They can be correctly explained by simple illustration of their simple conversions that may be made concrete by granulating their related translation algorithms.

4 Conclusions and future work

SAGEMoLiC, in its current state, is not a tutorial system and has to be used for an expert on formal language and automata theory to illustrate and explain notions to the students. This doesn't mean that the system is hard to use, but that the user has to have some experience with this theory to provide interesting examples of machines and language representations. For obtaining a tutorial system that could be used directly for students without knowledge about this field the system is being extended with a *passive* module, where the user will find very well selected built-in examples.

One interesting problem of graphical representation of state diagrams comes arise when new automata are generated from an original one. This is a very hard geometric problem related with the efficient two-dimensional representation of graphs. When a new automaton is being automatically generated *SAGEMoLiC* put states and transitions randomly in unused positions of the canvas. But during the whole process the user can change these positions by simple use of the mouse.

We agree, that in a certain way, here we have shown our very particular way to explain some theoretical concepts in the field of formal languages and automata theory. But what should be concluded here is that explaining relations between machine models and language representations, in the non-trivial cases, by simple granulation or debugging of specific translation algorithms is wrong, since that is restricted to the peculiarities of the given conversion, making the student (and in many opportunities the instructor too) forgetting the grade of generality of the studied equivalence relations and be focused (and confused) on properties of the algorithm itself, such as its terminating property, its correctness and its completeness. Of course, granulating translation algorithms is useful. But that was yet possible with the technology available some years ago. Today, the current sophisticated and very cheap software technology from simple personal computers makes it possible more accurate graphical educational tools that really help instructors to show and explain these relations to students.

References

[BE95] J. Barwise and J. Etchmendy. *Turing's Word 3.0.* CSLI Press, 1995.

[BLP+97] A. O. Bilska, K. H. Leider, M. Procopiuc, O Procopiuc, S. H. Rodger, J. R. Salemme, and E. Tsang. A Collection of Tools for Making Automata Theory and Formal Languages Come Alive. In *Twenty-eight SIGCSE Technical Symposium on Computer Science Education*, pages 15–19. ACM press, 1997.

[FB94] R.W. Floyd and R. Beigel. *The Language of Machines — An Introduction to Computability and Formal Languages.* Computer Science Press, 1994.

[GR99] E. Grammond and S. H. Rodger. Using *JFLAP* to Interact with Theorems in Automata Theory. In *Thirthieth SIGCSE Technical Symposium on Computer Science Education*, pages 336–340. ACM press, 1999.

[HR00] T. Hung and S. H. Rodger. Increasing Visualization and Interaction in the Automata Theory Course. Technical report, Duke University, 2000. *JFLAP* is available at `www.cs.duke.edu/~rodger/tools`.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[Kel95] D. Kelley. *Automata and Formal Languages — An Introduction.* Prentice Hall, 1995.

[Koz97] D. C. Kozen. *Automata and Computability.* Undergraduate Texts in Computer Science. Springer, 1997.

[LP81] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, 1981.

[Sud97] T. A. Sudkamp. *Languages and Machines: An Introduction to the theory of Computer Science.* Addison Wesley, 1997.

[Tay98] R. G. Taylor. *Models of Computation and Formal Languages.* Oxford University Press, 1998. *Deus ex Machina* N. Savoui `www.ics.uci.edu/~savoui/dem`.