Vizuelni i interaktivni alat za učenje konačnih automata

Nenad Jovanović, Srećko Stamenković, Dragiša Miljković
Fakultet tehničkih nauka, Univerzitet u Prištini
Kosovska Mitrovica, Srbija
nenad.jovanovic@pr.ac.rs, stamenkovic.srecko@gmail.com, dragisa.miljkovic@pr.ac.rs

Sažetak — **Softverski simulacioni alati za učenje uz pomoć računara postali su široko rasprostranjeni na svim nivoima obrazovanja. Prepoznati su kao alati za efikasno proučavanje složenih i apstraktnih sistema. Tradicionalnim učenjem teorije automata studenti ne mogu lako da vizualizuju teorijske konstrukcije, pa u ovom slučaju simulacioni sistemi predstavljaju odličan spoj teorijskog i praktičnog iskustva. U ovom radu predstavljen je alat koji je razvijen sa ciljem da, interaktivnom tehnikom, studentima pojednostavi i vizuelno prezentuje apstraktne koncepte i matematičke modele teorije automata. Konstruisanje konačnih automata moguće je u grafičkom editoru u obliku dijagrama stanja ili definisanjem funkcije prelaza pomoću tabele tranzicije. Simulator omogućava konverziju regularnog izraza u DFA, ili NFA, kao i simulaciju Tomsonovog algoritma.**

Ključne reči – teorija automata; konačni automati; simulacioni sistemi; edukativni alati;

I. UVOD

Teorija formalnih jezika se oslanja na apstraktne modele računarskih sistema koji se nazivaju „automati“. Automati su zapravo matematičke apstrakcije čije se izučavanje uglavnom zasniva na imaginaciji teorijskih koncepata. Razlog tome je kompleksnost izvođenja praktične nastave ili nedostatak odgovarajućih edukativnih alata. Studenti na studijama iz oblasti računarskih nauka se sa konceptom automata susreću kroz nastavne planove nekoliko različitih predmeta. Na primer, teorija formalnih jezika i automata, programski prevodioci, teorijsko računarstvo. Usled toga što se studenti na ovom predmetu po prvi put susreću sa određenim apstraktnim pojmovima, studenti kod takvih predmeta imaju problema sa razumevanjem i praćenjem nastavnog plana, dok za profesore nastava na ovim predmetima predstavlja izazovan zadatak.

Da bi se premostio jaz između teorije i njene praktične naučne primene, Goyal i Sachdeva u [1] predlažu različite strategije za „oživljavanje“ teorijskih koncepata koje uključuju korišćenje softverskih alata za vizualizaciju i interakciju. U cilju unapređenja tradicionalne nastave, Chesnevar, González i Maguitman u [2] predstavljaju nekoliko didaktičkih strategija koje uključuju korišćenje simulatora kao nastavnih pomagala. Simulacioni sistemi za učenje postali su uobičajena praksa u obrazovanju u okviru različitih oblasti nauke, a naročito u oblasti računarskog inženjerstva. Komparativna analiza simulacionih sistema za učenje programskih prevodilaca predstavljena je u [3], gde su prikazani i simulatori za učenje teorije formalnih jezika i automata.

U ovom radu opisan je softverski sistem za vizuelizaciju i simulaciju rada konačnih automata. Osnovna namena ovog sistema je za edukativne svrhe u vidu pomoćnog nastavnog sredstva. Rad je sistematizovan u pet poglavlja. U drugom poglavlju predstavljeni su neki značajniji trenutno dostupni simulacioni sistemi, različitih autora, koji su pogodni za učenje konačnih automata. Treće poglavlje opisuje osnovne karakteristike razvijenog softverskog alata. Rad sa simulacionim sistemom detaljnije je prezentovan u četvrtom poglavlju. U petom poglavlju je dat zaključak i izneti planovi za dalje usavršavanje alata.

II. PREGLED SIMULATORA POGODNIH ZA UČENJE KONAČNIH AUTOMATA

Ideja o upotrebi softverskih alata za simulaciju automata nije nova, pregled simulacionih alata koji obuhvata vremenski period od 50 godina predstavljen je u [4]. Međutim, poslednjih nekoliko godina značajnije se radi na razvoju sofisticiranijih simulacionih softvera za učenje teorije automata. Postoji veliki broj takvih simulatora, od jednostavnih sa prilično oskudnim funkcijama do višenamenskih koji nude moćne alate za interaktivnu edukaciju.

JFLAP (Java Formal Languages and Automata Package) je interaktivni grafički simulacioni alat napisan na Javi. Osnovne funkcije JFLAP-a odnose se na definisanje automata i gramatike. Konstruisanje automata se vrši pomoću grafičkog editora za crtanje koji omogućava rad sa nekoliko različitih vrsta automata kao što su konačni automati, Milijeva i Murova mašina, push-down automati, Tjuringova mašina. Definisanje jezika vrši se korišćenjem regularnih izraza ili beskonteksnih gramatika. Pored osnovnih funkcija, JFLAP obezbeđuje i druge funkcionalnosti, na primer, konverziju nedeterminističkog konačnog automata u deterministički, minimizaciju automata, transformaciju automata u odgovarajuće regularne gramatike i obrnuto, simulaciju procesa parsiranja [5]–[7].

Language emulator je softver razvijen u Javi koji omogućava manipulaciju regularnim izrazima, regularnom gramatikom, determinističkim konačnim automatima, nedeterminističkim konačnim automatima, nedeterminističkim konačnim automatima sa epsilon prelazima, Murovom i Milijevom mašinom. Ovaj simulacioni sistem omogućuje

studentima da kreiraju automate, definišu regularne gramatike, da izračunavaju unije i preseke definisanih gramatika, da vrše konverziju nedeterminističkog u deterministički konačni automat [8].

Seshat je simulacioni alat, baziran na web tehnologiji, koji studentima pruža mogućnost interakcije i eksperimentiranja sa najčešćim algoritmima leksičke analize i teorije automata. U pitanju su algoritmi za dizajn nedeterminističkog konačnog automata koji prepoznaje jezike definisane regularnim izrazima, za transformaciju nedeterminističkog konačnog automata u deterministički, za konstrukciju determinističkog konačnog automata korišćenjem regularnih izraza i algoritam za minimizaciju automata. Seshat je klijent-server aplikacija, gde je klijent dinamička web stranica, server je napisan na Python-u, a za razvoj korisničkog interfejsa korišćen je HTML, SVG, JavaScript, jQuery i CSS. Lekser i parser su izrađeni pomoću PLY (Python-ova implementacija Lex-a i Yacc-a) [9].

Automata Simulator je mobilna aplikacija koja obezbeđuje edukativno okruženje za simulaciju više tipova automata. Ova aplikacija je razvijena za mobilne telefone i tablete koji rade na Android operativnim sistemu, i zamišljena je kao pomoćno nastavno sredstvo za podršku studentima u procesu učenja. Studenti pomoću ove aplikacije mogu da dizajniraju i simuliraju konačne automate, push-down automate, Tjuringove mašine, kao i Murove i Milijeve mašine. Definisanjem osnovnih parametara izabranog tipa automata može se prikazati graf automata ili pokrenuti simulacija. Simulacija konstruisanog automata podrazumeva unos ulaznog niza, nakon čega se ispisuju koraci analize datog niza [10].

JCT (Java Computability Toolkit) predstavlja edukativno okruženje za konstrukciju i simulaciju konačnih automata i Tjuringovih mašina. JCT se sastoji od dva web-bazirana grafička okruženja razvijena u Javi. Konačni automati i Tjuringove mašine se konstruišu grafički u okviru različitih korisničkih interfejsa. Zadavanjem proizvoljnog ulaza, studenti mogu da prate kako automat radi, korak po korak, uz neposredne vizuelne povratne informacije [11].

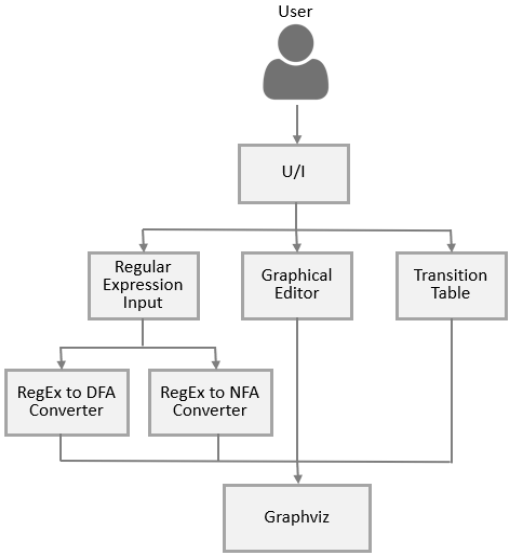
Robot-based Automata Simulator predstavlja simulacioni sistem za učenje teorije automata, razvijen kombinacijom Jave i robotskih tehnologija [12], [13]. Simulator može komunicirati sa robotom koji je dizajniran za ovu svrhu korišćenjem Lego NXT seta. Komunikacija robota i simulatora ostvaruje se između Java okruženja i LeJOS softvera. Studenti konstruišu automat u okviru grafičkog interfejsa sistema, a zatim ga prosleđuju robotu koji može simulirati tranzicije automata. Svrha korišćenja simulatora i robota je učenje teorije automata kroz igru, posmatranjem kretanja robota.

jFAST (Java Finite Automata Simulation Tool) je robusna i intuitivna aplikacija implementirana u Javi, a koja korisniku pruža mogućnost da na brz i jednostavan način konstruiše i modifikuje automate [14]. Kreiranje automata je veoma jednostavno u okviru intuitivnog grafičkog korisničkog interfejsa. jFAST pruža modularni pristup za formiranje složenih automata primenom tako zvanih pod-automata. jFAST podržava nekoliko različitih tipova automata i to: deterministički i nedeterministički konačni automat, push-down automat, Tjuringovu mašinu i jednostavni apstraktni model automata. Radno okruženje za sve podržane tipove automata je isto.

III. OSNOVNE KARAKTERISTIKE SIMULATORA KONAČNIH AUTOMATA

Simulacioni sistem je realizovan u Java programskom jeziku kako bi se omogućila njegova prenosivost na svaki sistem koji podržava Java Virtuelnu Mašinu. Sistem je dizajniran pomoću Java Swing biblioteke za razvoj grafičkog korisničkog interfejsa. Sistema je dizajniran u skladu sa objektno-orijentisanim pristupom rešavanju problema, čime je ostvarena lakoća održavanja i prilagođavanja. Dodavanje novih komponenti u sistem je jednostavno, uz maksimalno iskorišćavanje već definisanih modula i bez potrebe za njihovim izmenjivanjem. Interfejs sistema je dizajniran na engleskom jeziku, ali je ostavljena mogućnost za njegovu internacionalizaciju dodavanjem još podržanih jezika. Sistem je dostupan kao desktop aplikacija koja omogućava korisniku da brzo i jednostavno dizajnira, menja i grafički simulira konačne automate. Dodatno, moguće je i čuvanje dizajniranih automata kao i otvaranje i izmenjivanje ranije sačuvanih. U pitanju je interaktivna aplikacija sa grafičkim korisničkim interfejsom koji omogućava da se proces simulacije može pratiti korak po korak. Osnovna namena ovog simulatora je da unapredi kvalitet tradicionalne nastave tako što omogućuje studentima platformu za interaktivno eksperimentisanje, pri čemu dobijaju trenutne i pouzdane povratne informacije o rezultatima eksperimenta.

Struktura simulacionog softvera prikazana je na Sl. 1. Ovaj alat se sastoji od osam osnovnih komponenata: korisnički ulazno/izlazni interfejs, komponenta za unos regularnih izraza, konverter regularnih izraza u deterministčke konačne automate, konverter regularnih izraza u nedeterminističke konačne automate, grafička komponenta za konstruisanje konačnih automata – grafički editor, komponenta za kreiranje tabele tranzicije i Graphviz – vizualizator.



Slika 1. Struktura simulacionog sistema.

Na osnovu prikazane strukture simulatora, može se videti da postoje tri različita pristupa za konstruisanje konačnih automata:

1) Prvi pristup je konstruisanje automata na osnovu definisanog regularnog izraza. Regularni izraz se može konvertovati u deterministički ili nedeterministički konačni automat, po izboru korisnika.

2) Drugi pristup jeste crtanje automata pomoću grafičkog editora, na platnu uz korišćenje ponuđenih objekata – simbola preko jednostavnih operacija miša.

3) Treći pristup je definisanje funkcije prelaza pomoću tabele tranzicije.

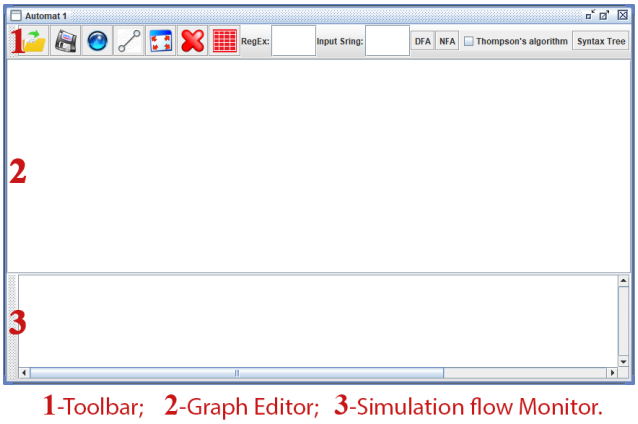
Izgled osnovnog korisničkog prozora simulacionog softvera prikazan je na Sl. 2. Aplikacija se sastoji iz tri osnovna elementa:

1) Paleta alata koja nudi niz mogućnosti za manipulaciju konačnim automatima. Interesantne opcije su prikaz rada Tomsonovog algoritma i formiranje sintaksnog stabla.

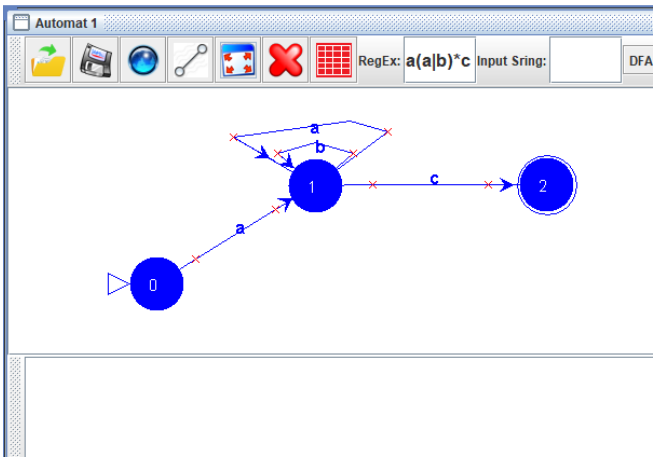
2) Grafički editor zauzima srednji, najveći deo prozora i predstavlja polje za crtanje automata.

3) Monitor toka simulacije nalazi se u donjem delu prozora aplikacije i koristi se za tekstualni opis scenarija rada konačnog automata.

Za crtanje automata koriste se četiri opcije na paleti alata. Izborom opcije *Stanje* mogu se dodavati čvorovi grafa, to jest nova stanja automata, i to klikom levim tasterom miša u okviru polja grafičkog editora. Klikom desnim tasterom miša na određeni čvor pojavljuju se opcije za definisanje *Startnog* ili *Konačnog* stanja. Početno stanje biće jasno obeleženo ulaznom strelicom, dok će čvor konačnog stanja biti uokviren kružnom linijom. Opcija *Tranzicija* se odabira kada je potrebno definisati prelaze između odgovarajućih stanja konačnog automata. Nakon crtanja prelaza, neophodno je uneti i simbol kojim se obezbeđuje data tranzicija. U slučaju kada postoji veliki broj stanja i prelaza, poželjno je iskoristiti opciju *Pomeri* kako bi se elementi automata prasporedili zarad bolje preglednosti. Pomeranje čvorova ne utiče na definisane prelaze i oni se ne raskidaju. U slučaju kada je potrebno obrisati nacrtana stanja i prelaze koristi se opcija *Brisanje*.



Slika 2. Izgled prozora simulacionog softvera.



Slika 3. Definisanje DFA na osnovu izraza $a(a|b)^*c$.

Pored navedenih opcija, na paleti alata nalazi se i opcija *Tabela* koja se koristi za kreiranje konačnog automata pomoću tabele prelaza. Na paleti su još i dva tekstualna polja, prvo služi za unos regularnog izraza, a drugo za unos ulaznog stringa za koji će se ispitati rad automata. Njih slede dva dugmeta, pod imenom DFA i NFA, čijim se izborom vrši konvertovanje unetog regularnog izraza u deterministički (DFA) ili nedeterministički konačni automat (NFA), respektivno.

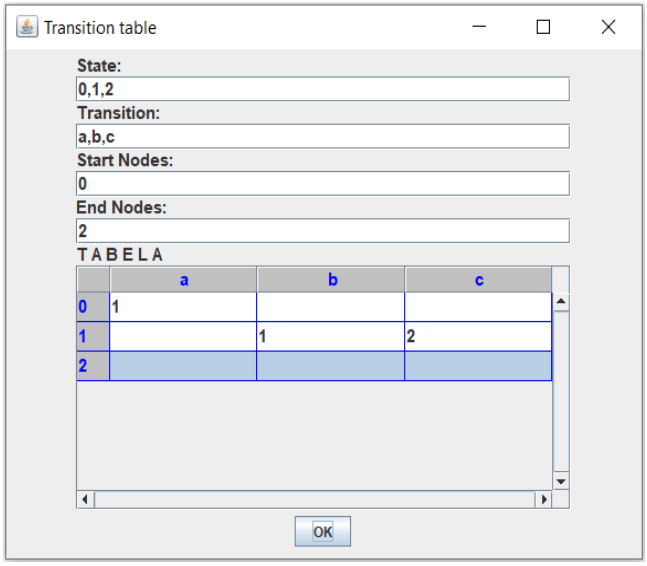
Za transformaciju regularnog izraza u ekvivalentni nedeterministički konačni automat simulator koristi Tomsonovu konstrukciju [15]. U pitanju je algoritam koji rastavlja regularni izraz na njegove sastavne podizraze, od kojih potom se na osnovu skupa pravila konstruiše NFA. Da bi se prikazali svi koraci u procesu konvertovanja, tj. za simulaciju Tomsonovog algoritma, potrebno je čekirati opciju Thompson's algorithm. Poslednja opcija na paleti alata obezbeđuje konstrukciju sintaksnog stabla za navedeni regularni izraz.

IV. Opis rada simulacionog sistema

Konačne automate u simulacionom okruženju definišemo na tri načina navedena u prethodnom poglavlju. Na Sl. 3 prikazano je konstruisanje konačnog automata na osnovu regularnog izraza $a(a|b)^*c$.

Automat naveden iznad se može dobiti i na direktan način – crtanjem dijagrama stanja pomoću alata koji su dostupni na paleti, a na način koji je objašnjen u prethodnom poglavlju. Konstrukcija automata korišćenjem tabele prelaza je pogodna kada se radi sa automatima koji imaju mali broj stanja i prelaza. Da bi se uz pomoć tabele prelaza kreirao isti automat kao onaj na Sl.3, potrebno je popuniti tabelu za zadate skupove $\Sigma=\{a,b,c\}$, $Q=\{1,2,3\}$, $I=\{0\}$, $F=\{2\}$ kao što je to prikazano na Sl. 4.

Simulator konačnih automata pruža i funkcionalnost konverzije definisanih DFA u NFA, kao i konverziju regularnih izraza u njima ekvivalentne NFA, izborom odgovarajuće opcije na paleti alata. Za ovu konverziju koristi se Tomsonov algoritam koji je definisan skupom pravila prikazanih na Sl.5.

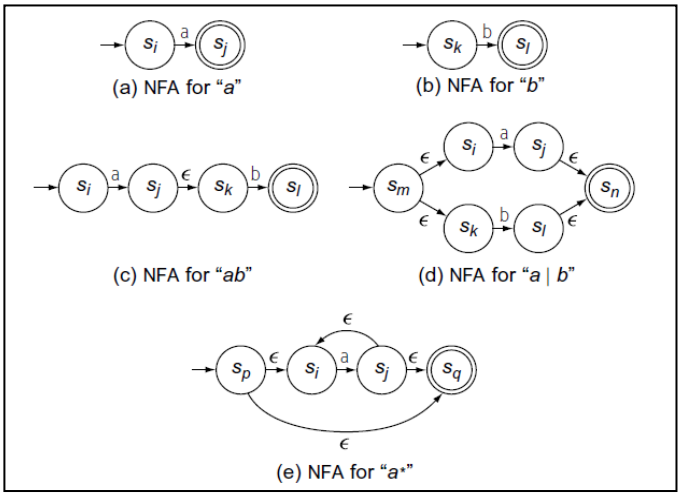


Slika 4. Definisanje DFA na osnovu tabele tranzicije.

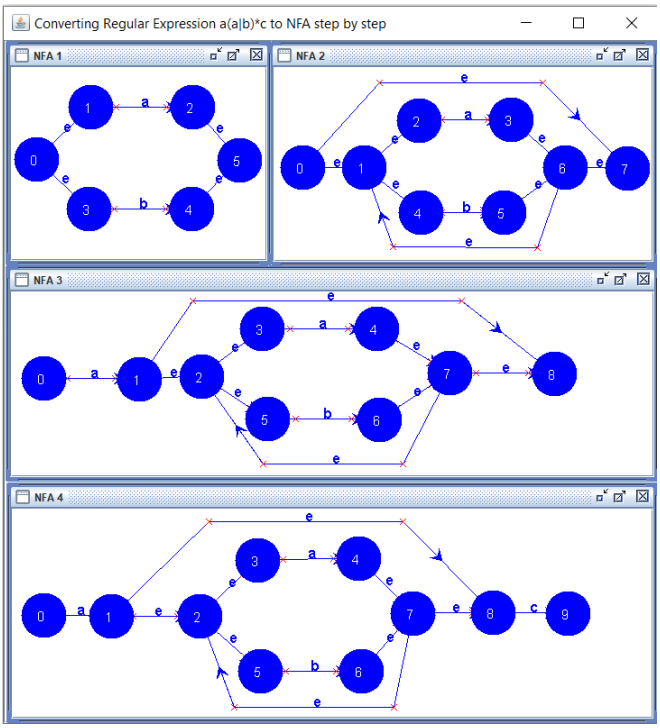
Simulacioni sistem obezbeđuje prikaz svih primenjenih transformacija zadatog regularnog izraza u ekvivalentni NFA, korak po korak, ukoliko se čekira ta opcija. Koraci transformacije za regularni izraz *a*(*a*|*b*)**c* prikazani su na Sl.6.

Konačni automati se koriste za prepoznavanje jezika tipa 3 Chomsky hijerarhije, odnosno regularnih jezika. Za ulazni niz *w* kažemo da je prepoznat automatom ako automat prevodi iz početnog u neko od njegovih krajnjih stanja, odnosno ako je $P(q0,w) \in F$. Gde *P* predstavlja pravila preslikavanja, *q0* početno stanje i *F* skup završnih odnosno krajnjih stanja.

Za simulaciju rada konstruisanog automata potrebno je uneti ulazni string u odgovarajuće tekstualno polje. Pritiskom na taster Enter započinje proces simulacije. Pored vizuelne prezentacije u polju grafičkog editora, tok simulacije se može pratiti i u tekstualnom obliku u okviru monitora toka simulacije. Prikaz procesa simulacije za ulazni string *aabbc* prikazan je na Sl. 7.

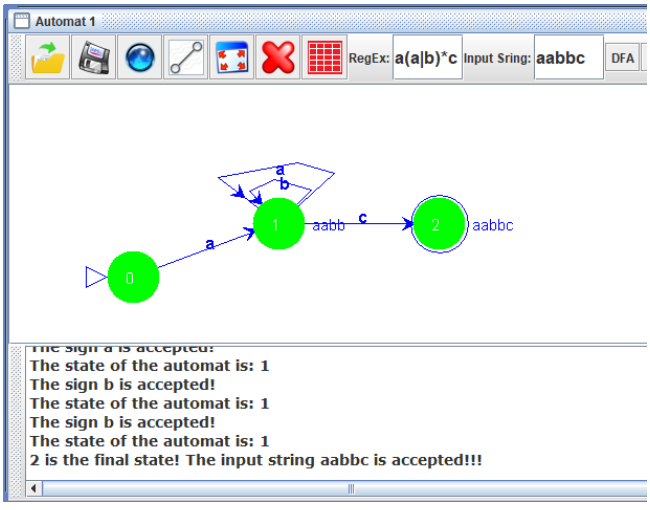


Slika 5. Skup pravila Tomsonovog algoritma koja se primenjuju za odgovarajuće operatore regularnih izraza. (Preuzeto iz [15]).



Slika 6. Konvertovanje regularnog izraza *a*(*a*|*b*)**c* u NFA, korak po korak.

Kada je pokrenuta simulacija rada automata, sistem za sve simbole ulaznog stringa prikazuje postupak analize korak po korak, pri čemu se tekuće stanje automata predstavlja crvenom bojom i posebno naznačuje simbol koji se analizira u tom trenutku. Ukoliko je simbol prihvaćen, tj. ako automat ima prelaz za taj simbol, to stanje se obeležava zelenom bojom, a automat prelazi u sledeće stanje. Istovremeno sa tim, korisnik na ekranu dobija u tekstualnom obliku informaciju o trenutnom stanju koje se analizira, a na kraju analize i informaciju o tome da li je automat prepoznaje kompletan ulazni string. U slučaju kada automat ne prepozna uneti string, na ekranu se ispisuje poruka o grešci.



Slika 7. Simulacija rada konstruisanog automata za ulazni string *aabbc*.

V. ZAKLJUČAK

U ovom radu predstavljen je simulacioni sistem za učenje konačnih automata. Ovaj alat razvijen je sa ciljem da studentima pomogne u procesu učenja i razumevanju teorije automata omogućujući vizualizaciju teorijskih konstrukcija kroz interaktivni grafički interfejs.

Definisanje automata u simulatoru se može izvršiti na više različitih načina: korišćenjem jednostavnog drag-and-drop interfejsa, preko tabele prelaza i na osnovu regularnog izraza. Nakon konstrukcije automata, moguće je pokrenuti simulaciju rada za proizvoljan ulazni string. Korisnik dobija povratne informacije o svakom koraku simulacije u grafičkom i tekstualnom obliku. Pored simulacije rada konačnih automata, dodatne funkcije sistema su: konverzija regularnih izraza u DFA, kao i u NFA, simulacija Tomsonovog algoritma i formiranje sintaksnog stabla analizom regularnog izraza.

Simulacioni sistem je još uvek u fazi razvoja i može biti značajno unapređen. Dalji koraci u pogledu usavršavanja simulacionog sistema jesu u smeru proširivanja njegovih funkcionalnosti dodatnim tipovima automata. Takođe, predviđeno je proširivanje funkcionalnosti sistema tako da se obuhvate dodatne oblasti teorije formalnih jezika. Konačno, u toku je i izrada Web aplikacije sa višejezičnim interfejsom preko koje će alat biti dostupan za besplatno korišćenje svim zainteresovanim pojedincima i grupama.

LITERATURA

[1] M. Goyal and S. Sachdeva, "Enhancing Theory of Computation teaching through integration with other courses", International Journal of Recent Trends in Engineering, 2009, 1(2), pp. 137-140.

[2] C. I. Chesnevar, M. P. González and A. G. Maguitman, "Didactic strategies for promoting significant learning in formal languages and automata theory". ACM SIGCSE Bulletin, 2004, 36(3), pp. 7-11.

[3] S. Stamenković and N. Jovanović, "Comparative analysis of simulation system for teaching compilers". BizInfo (Blace), 2019, 10(1).

[4] P. Chakraborty, P. C. Saxena and C. P. Katti, "Fifty years of automata simulation: a review." ACM Inroads, 2011, 2(4), pp. 59-70.

[5] M. Procopiuc, O. Procopiuc and S. H. Rodger, "Visualization and interaction in the computer science formal languages course with JFLAP". In Frontiers in Education FIE'96 26th Annual Conference, 1996, pp. 121-125.

[6] S. H. Rodger and T. W. Finley, "JFLAP: an interactive formal languages and automata package". London: Jones & Bartlett Publishers, 2006.

[7] S. H. Rodger, E. Wiebe, K. M. Lee, C. Morgan, K. Omar and J. Su, "Increasing engagement in automata theory with JFLAP". In ACM SIGCSE Bulletin, 2009, 41(1), pp. 403-407.

[8] L. F. M. Vieira, M. A. M. Vieira and N. J. Vieira, "Language emulator, a helpful toolkit in the learning process of computer theory". In ACM SIGCSE Bulletin, 2004, 36(1), pp. 135-139.

[9] Á. Arnaiz-González, J. F. Díez-Pastor, I. Ramos-Pérez and C. García-Osorio, "Seshat—a web-based educational resource for teaching the most common algorithms of lexical analysis". Computer Applications in Engineering Education, 2018, 26(6), 2255-2265.

[10] T. Singh, S. Afreen, P. Chakraborty, R. Raj, S. Yadav and D. Jain, "Automata Simulator: A mobile app to teach theory of computation". Computer Applications in Engineering Education, 2019, [online]. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.22135 [Accessed 11 Aug. 2019].

[11] M. B. Robinson, J. A. Hamshar, J. E. Novillo and A. T. Duchowski, "A Java-based tool for reasoning about models of computation through simulating finite automata and Turing machines". In ACM SIGCSE Bulletin, 1999, 31(1), pp. 105-109.

[12] M. Hamada and S. Sato, "A game-based learning system for theory of computation using Lego NXT Robot". Procedia Computer Science, 2011, 4, 1944-1952.

[13] M. Hamada and S. Sato, "A learning system for a computational science related topic". Procedia Computer Science, 2012, 9, 1763-1772.

[14] T. M. White and T. P. Way, "jFAST: A java finite automata simulator". In ACM SIGCSE Bulletin, 2006, 38(1), 384-388.

[15] K. D. Cooper and L. Torczon, "Engineering a Compiler, Second Edition". USA: Elsevier, Inc, 2012.

ABSTRACT

Simulation tools for computer-assisted learning are widespread at all levels of education. They are recognized as tools for effective study of complex and abstract systems. Traditional learning of automata theory is not easy for students to visualize theoretical constructs, so in this case, simulation systems are a great blend of theoretical and practical experience. This paper presents a tool developed with the aim of simplifying and visually presenting abstract concepts and mathematical models of automata theory to students. Constructing finite automata is possible in a graph editor in the form of a state diagram or by defining a transition function using a transition table. The simulator allows converting regular expressions to DFA or NFA, as well as simulating the Thompson algorithm.

A VISUAL AND INTERACTIVE TOOL FOR LEARNING FINITE AUTOMATA

Nenad Jovanović, Srećko Stamenković, Dragiša Miljković