Часопис из области економије, менаџмента и информатике
БизИнфо (БЛАЦЕ)
Година 2019, волумен 10, број 2, стр. 1-23

Journal of Economics, Management and Informatics
BizInfo (BLACE)
Year 2019, Volume 10, Number 2, pp. 1-23

Оригиналан научни рад/ Original scientific paper
УДК/UDC: 004.4'4:[371.214.11+378
004.4'4
doi:10.5937/bizinfo1902001S

KOMPARATIVNA ANALIZA SIMULACIONIH SISTEMA ZA UČENJE PROGRAMSKIH PREVODILACA

COMPARATIVE ANALYSIS OF SIMULATION SYSTEM FOR TEACHING COMPILERS

Srećko Stamenković[1]
Univerzitet u Prištini, Fakultet tehničkih nauka, Kosovska Mitrovica, Srbija

Nenad Jovanović
Univerzitet u Prištini, Fakultet tehničkih nauka, Kosovska Mitrovica, Srbija

Sažetak: *U ovom radu izvršena je analiza simulacionih sistema za učenje programskih prevodilaca, predstavljene su njihove tehničke karakteristike, teme koje obrađuju, benefiti njihovog korišćenja u nastavi, kao i prednosti i mane svakog analiziranog simulatora. Na osnovu ovih informacija čitalac lako može da donese odluku o tome koji simulator je za njegove potrebe najpogodniji. Simulatori su u obrazovanju prepoznati kao alati za efikasno proučavanje kompleksnih i dinamičnih sistema. Veliki broj teorijskih apstraktnih koncepata iz različitih faza programskog prevođenja može se ilustrovati pomoću simulatora. Simulacioni sistemi u oblasti programskih prevodilaca predstavljaju odličan spoj teorijskog i praktičnog iskustva, i kao takvi doprinose unapređenju nastavnog procesa.*
Ključne reči: *simulacioni sistemi, programski prevodioci, teorija automata, formalni jezici*

Abstrakt: *This paper analyses simulation systems for teaching compilers, represents their technical characteristics and topics addressed, their benefits in the teaching process, and the advantages and disadvantages of each simulator analyzed. Based on this information, a reader can easily decide*

[1] stamenkovic.srecko@gmail.com

which simulator is the most suitable for his needs. The usage of simulators in education has been recognized as an effective tool for exploration of complex and dynamic systems. A simulator can illustrate a large number of abstract theoretical concepts from the various phases of program translation. Simulation systems in the field of compilers stand for a perfect combination of theory and practical experiences, and as such contribute to the improvement of the teaching process.
Key words: *simulation systems, compilers, automata theory, formal languages*

1. UVOD

Nastavni planovi na studijama iz oblasti računarskih nauka sadrže veći broj predmeta čije se izučavanje zasniva na imaginaciji teorijskih koncepata, zbog kompleksnosti izvođenja praktične nastave ili nedostatka odgovarajućih edukativnih alata. Studenti treba da razviju sposobnost apstraktnog razmišljanja o određenim računarskim procesima kako bi uspešno savladali predmet izučavanja. Međutim, mnogi studenti, kod takvih predmeta, imaju problema sa razumevanjem i praćenjem nastavnog plana, pa je procenat neuspeha na ispitima obično vrlo visok (Knobelsdorf, Kreitz and Böhne, 2014). Ovo posebno važi za predmete koji obrađuju teme koje se odnose na formalne jezike, teoriju automata i konstrukciju programskih prevodioca. Obično u nastavi kod ovih predmeta, studenti dobijaju samo površno znanje kroz teorijske koncepte. Tradicionalnim učenjem ne mogu lako da vizualizuju teorijske konstrukcije (Yalagi and Dixit, 2017). Navedene okolnosti su motivisale razvoj brojnih simulatora kao obrazovnih alata, koji omogućavaju studentu da jednostavnije savlada mnoge teme koje su proučavane na tradicionalan način. Postoje brojne studije koje ukazuju na pozitivan uticaj upotrebe simulacionih sistema, kod studenata, neke od njih će biti predstavljene u ovom radu.

Cilj ovog istraživanja je da se izvrši analiza trenutno dostupnih simulacionih sistema u oblasti programskih prevodioca, ukaže na sve prednosti koje donosi njihovo korišćenje u nastavi, kao i da se od analiziranih postojećih rešenja predlože najefikasniji modeli. U drugom poglavlju ukratko su predstavljene teorijske osnove programskih prevodioca, kako bi se definisali osnovni pojmovi koji će se koristiti u ostatku rada. Treće poglavlje se bavi istraživanjima koja detaljno analiziraju efekte korišćenja simulacionih alata. U četvrtom poglavlju su predsavljena brojna rešenja koja nude različiti autori u svojim istraživačkim radovima. Dok su u petom poglavlju detaljnije analizirani odabrani simulacioni alati.

2. PROGRAMSKI PREVODIOCI

Planom i programom akademskih studija u oblasti računarskih nauka obavezno je obuhvaćen i predmet koji proučava programske prevodioce. Od fakulteta do fakulteta sadržaj ovog predmeta varira, ali u suštini on obuhvata sledeće oblasti (ili bar neke od navedenih):

- Teorija formalnih jezika (opis jezika, elementi jezika, operacije nad jezicima, formalne gramatike...);
- Teorija automata (klase automata, osnovni tipovi automata, prepoznavanje jezika, preslikavanje regularnih izraza u konačne automate...);
- Leksička analiza (leksemi, tokeni, šabloni, realizacija leksičkog analizatora...);
- Sintaksna analiza (algoritmi za sintaksnu analizu, sintaksna stabla, realizacija parser generatora...);
- Semantička analiza (semantička pravila, tabele simbola);
- Generatori međukoda (apstraktno sintaksno stablo, troadresni međukod, optimizacija međukoda...);
- Generatori koda (preslikavanje međukoda u odgovarajući asemblerski jezik, načini adresiranja, upravljanje memorijom).

Da bi se matematički proučavali programski jezici, potreban je mehanizam da se precizno opišu. Svakodnevni govorni jezik je neprecizan i dvosmislen, tako da su neformalni opisi govornog jezika često neadekvatni. Zbog toga se koristi formalni opis programskog jezika koji predstavlja njegovu standardizaciju. Noam Chomsky osnivač teorije formalnih jezika, definisao je četiri tipa gramatike, i to: gramatike tipa 0, gramatike tipa 1 ili konteksne, gramatike tipa 2 ili beskonteksne i gramatike tipa 3 ili regularne gramatike (Linz, 2012). Odnos jezika definisanih navedenim tipovima gramatika može se predstaviti sledećom relacijom: $L(3) \subset L(2) \subset L(1) \subset L(0)$. Iz ove relacije može se videti da su gramatike tipa 0 najopštije, što znači da se sva algoritamska preslikavanja mogu opisati ovim gramatikama, dok gramatike tipa 3 pokrivaju najuži skup jezika (Jäger and Rogers, 2012).

Teorija formalnih jezika se oslanja na apstraktne modele računskih sistema nazvanih „automati“. Automati su zapravo matematičke apstrakcije, teoretske a ne realne mašine (Fitch and Friederici, 2012). Na primer, Tjuringova mašina uključuje, kao deo svoje matematičke definicije, radnu memoriju u vidu beskonačne trake, tako da je nemoguće konstruisati pravu Tjuringovu mašinu. Automati ovog tipa mogu se koristiti za prepoznavanje jezika tipa 0. Za prepoznavanje jezika tipa 3, odnosno regularnih jezika koriste se konačni automati. Ovi automati se dele na determinističke i nedeterminističke

konačne automate. Deterministički konačni automat je automat koji prihvata konačne nizove simbola, tako što za svako stanje, za određeni ulazni simbol, jednoznačno definiše sledeće stanje automata. Kod nedeterminističkog konačnog automata za svaki par stanja i ulaznog simbola može postojati više od jednog sledećeg stanja (Indu, 2016).
Regularnim izrazima definišu se leksički elementi koji se prepoznaju leksičkim analizatorom. Leksički analizator čita ulazni niz izvornog programa, grupiše znakove u leksičke jedinice koje se nazivaju leksemi, i generiše odgovarajuće izlazne tokene (Aho et al., 2007). Token se sastoji od dve komponente, ime tokena i vrednosti atributa. Dobijeni niz tokena iz leksičkog analizatora dovodi se na ulaz sintaksnog analizatora. Funkcija sintaksnog analizatora (parsera) je generisanje sintaksnog stabla. U zavisnosti od toga da li se analiza vrši odozgo naniže ili odozdo naviše razlikujemo top-down i bottom-up analizu. Top-down parseri su jednostavniji za konstrukciju, dok bottom-up parseri mogu da rade sa višom klasom gramatike, tako da softverski alati za generisanje parsera češće koriste analizu odozdo naviše.

Najčešće korišćeni analizatori, koji se realizuju primenom top-down analize, su LL-1 analizatori, kod kojih se predikcija vrši na osnovu jednog karaktera u ulaznom nizu. Da bi predikcija bila moguća, beskonteksna gramatika treba da bude definisana tako da u skupu pravila postoji najviše jedna raspoloživa smena za jedan ulazni simbol i jedan neterminal (Stanković, Stojković and Tošić, 2018). Najrasprostranjeniji tip bottom-up parsera zasniva se na konceptu LR parsiranja. LR parseri mogu biti konstruisani tako da prepoznaju praktično sve jezike koji se mogu definisati beskonteksnim gramatikama.

Za semantičku analizu koriste se tabele simbola, u kojima se čuvaju informacije koje se prikupljaju postepeno u fazama analize. Podaci u tabeli simbola sadrže informacije o identifikatoru kao što je niz znakova (ili leksem), njegov tip, položaj u skladištu i sve druge relevantne informacije. Tabele simbola obično moraju podržavati višestruke deklaracije istog identifikatora unutar programa (Aho et al., 2007).

Nakon završetka prve faze, tj. faze analize programskog prevođenja, prelazi se na drugu fazu, tj. fazu sinteze. Glavni zadatak ove faze je generisanje ciljnog koda. Međutim, generisanju koda prethodi generisanje međukoda. Prednost ovakvog postupka je ta što je međukod bliži mašinskom kodu ali ipak nezavisan od mašine, čime je obezbeđena prenosivost, kao i ta što se na ovom nivou mogu koristiti programi za optimizaciju koji su nezavisni od ciljne mašine. Međukod može imati oblik apstraktnog sintaksnog stabla, postfiksne (poljske) notacije ili prefiksne notacije i troadresnog koda.

Na kraju procesa programskog prevođenja, na ulazu generatora koda dovodi se generisana reprezentacija izvorng programa u obliku međukoda, kao i

odgovarajuće informacije iz tabele simbola. Informacije u tabeli simbola koriste se da bi se odredile adrese određenih objekata u međokodu. Na izlazu generatora koda dobija se ciljni program. Kao i u slučaju generisanja međukoda, i ovde izlaz može imati različite oblike, kao što su apsolutni mašinski kod, relativni mašinski kod ili kod asemblerskog jezika (Su and Yan, 2012).

Iz ovog kratkog teorijskog pregleda može se videti da je prevođenje programa veoma složen proces koji se sastoji iz više faza, od leksičke analize do optimizacije koda. Svaka od ovih faza se može proučavati nezavisno. Autori simulacionih sistema, uglavnom nude edukativne alate koji simuliraju jednu ili nekoliko povezanih faza. Veoma je kompleksno razviti sveobuhvatni alat koji će simulirati kompletan proces programskog prevođenja.

3. KORIŠĆENJE SIMULACIONIH ALATA U NASTAVI

Simulatori su, kako na akademskim tako i na strukovnim studijama, prepoznati kao alati za efikasano i efektivano podučavanje složenih i dinamičnih sistema. Softver za simulaciju omogućava studentima da interaktivno eksperimentišu, pružajući trenutnu i pouzdanu povratnu informaciju o uspešnosti eksperimenta. Na ovaj način studenti dobijaju priliku da isprobaju različite opcije i trenutno procene stečeno teorijsko znanje. Dakle, može se reći da interaktivne simulacije predstavljaju moćno sredstvo za pobuđivanje naučno-istraživačkog razmišljanja kod studenata.

Prema Nahvi (1996) simulatori koji se koriste u nastavi moraju da zadovolje određene zahteve. Trebali bi da budu intuitivni i jednostavni za korišćenje. Takođe, bi trebali da budu specijalizovani, a ne alati opšte namene, čime se obezbeđuje njihova efikasnost i lakoća pri radu. I ono što je veoma važno, a što ističu i drugi autori, da simulacioni alat ne treba da bude osnovno nastavno sredstvo, već pomoćno u službi klasične teorijske nastave. Slični zaključci se mogu naći u radu (Taher and Khan, 2015), gde se naglašava da simulacija sama po sebi nije veoma efikasna u unapređenju učenja studenata, ali postaje izuzetno moćno nastavno sredstvo kada se koristi zajedno sa tradicionalnom nastavom. Takav način pristupa autori nazivaju hibridnom odnosno kombinovanom strategijom.

Lindgren i Schwartz (2009) u svom radu uvode četiri efekta učenja kako bi razjasnili pozitivne aspekte upotrebe simulacija: superiornost slike, zapažanje, strukturiranje i podešavanje. Ljudi imaju impresivnu memoriju za vizuelne informacije i prostornu strukturu, što znači da korišćenje vizuelnih komponenti može imati samo benefit u korist edukacije. Karakteristika perceptualnog učenja je sposobnost da se više uočava u datoj situaciji. Eksperti u nekoj situaciji mogu primetiti neke finese koje početnici

jednostavno ne vide. Simulacioni sistemi treba da omoguće upravo to, da studenti zapažaju i ono što samostalnim učenjem ne mogu.
Pouzdane dokaze da kompjuterske simulacije mogu unaprediti tradicionalne načine edukacije pružaju Rutten, Van Joolingen and Van der Veen (2012). Ovaj članak razmatra veliki broj eksperimentalnih istraživanja, koja su objavljena u vremenskom periodu od jedne decenije, o efektima koji se postižu korišćenjem kompjuterskih simulacija u obrazovanju. Rad se fokusira na dva pitanja: kako korišćenje kompjuterskih simulacija može poboljšati tradicionalno obrazovanje i na koji način treba upotrebiti kompjuterske simulacije za najbolje rezultate u pogledu poboljšanja procesa i ishoda učenja. Sve analizirane studije koje upoređuju uslove rada sa i bez simulacija, pokazuju pozitivne rezultate u korist upotrebe simulacija za poboljšanje tradicionalnih načina učenja.

Simulacioni alati danas su dostupni skoro za svaku oblast nauke. Iskustvo o upotrebi simulacionog softvera za oblast obnovljivih izvora energije predstavljeno je i razmatrano u (Witzig et al., 2016). Utvrđeno je da je simulacija korisna za razumevanje podataka koji su inače nedostupni, tako da teme koje se odnose na solarne i obnovljive izvore energije postaje opipljivije. Veliki broj simulacionih sistema se koristi i u oblasti elektronskih nauka. Zhang and Jie (2018) smatraju da je teorija analogne elektronske tehnologije veoma apstraktna i teško razumljiva, i zbog toga predlažu uvođenje simulacionog softvera za učenje. U članku se opisuju karakteristike simulacionog softvera i način na koji on pomaže u prevazilaženju teškoća u nastavi. Ovaj edukativni alat, jednostavnog i prilagodljivog interfejsa, omogućava bolje razumevanje tema iz oblasti analogne elektronike i poboljšava efikasnost nastave. Jovanović N., Jovanović Z. and Jevremović (2016) analizirali su veći broj simulacionih sistema iz oblasti računarskih mreža. Web bazirani simulatori za učenje računarskih mreža predstavljeni su u (Jovanović et al., 2012; Jovanović, Popović and Jovanović, 2009), dok su simulatori koji koriste spektralnu teoriju grafova za proučavanje mreža opisani u (Jovanović and Zakić, 2018; Jovanović, Zakić and Veinović, 2016). Edukativni matematički alat koji svakako zavređuje pažnju je Wolfram Mathematica (Wolfram, 2003). U početku, softver Mathematica se uglavnom koristio u matematici, fizici i inženjerstvu, dok je tokom godina razvoja, Mathematica postala značajan alat u mnogim oblastima nauke. To je jedana od najvećih aplikacija ikada razvijenih, sadrži širok spektar algoritama i važnih tehničkih inovacija. Efekat učenja teorije verovatnoće uz pomoć simulacionog softvera proučavan je u (Koparan and Yilmaz, 2015). Istraživanje je vršeno na grupu od 55 studenata, pri čemu je u njihovoj nastavi korišćen dinamički statistički softver TinkerPlots (Konold and Miller, 2004). Dobijeni nalazi su pokazali da je učenje verovatnoće zasnovano na simulaciji unapredilo veštinu predviđanja i donošenja zaključka, odnosno ovakav vid učenja je pozitivno uticao na uspeh studenata. Edukativni

kompjuterski sistem sa web-baziranim simulatorom, dizajniran da pomogne u nastavi i učenju računarske arhitekture, predstavili su autori u (Djordjevic, Nikolic, and Milenkovic, 2005). U pitanju je fleksibilno, web bazirano, edukativno okruženje osmišljeno da pomogne u podučavanju i učenju računarske arhitekture, omogućavajući vizualizaciju funkcionisanja računarskog sistema sa različitim nivoima detalja. Primena kompjuterskih simulacije u nastavi mikrobiologije razmatrana je u (Huppert, Lomask and Lazarowitz, 2002). Simulacioni alat Guess za izučavanje teorije grafova predstavljen je u (Adar, 2006). U oblasti genetike Soderberg and Price (2003), predlažu edukativni softver EVOLVE.

Kao što se iz napred navedenog vidi, simulacioni sistemi se koriste u gotovo svim naučnim oblastima, gde beleže pozitivne rezultate u pogledu unapređenja tradicionalne nastave. Predstavljanje simulacionih alata u oblasti formalnih jezika, teorije automata i konstrukcije programskih prevodilaca ovde je namerno izostavljeno, jer se pregledom takvih pedagoških alata bavi sledeće poglavlje.

4. SIMULACIONI SISTEMI U OBLASTI PROGRAMSKIH PREVODILACA

Nastava programskih prevodioca za profesore predstavlja izazovan zadatak, s obzirom da se studenti na ovom predmetu, po prvi put susreću sa određenim apstraktnim pojmovima za čije objašnjenje je potrebna dobra matematička osnova. Teme koje se izučavaju su od suštinskog značaja za nastavne programe iz oblasti računarskih nauka, pa je studentima i te kako u interesu da ih sa razumevanjem savladaju.

Na osnovu dugogodišnjeg nastavnog iskustava, na studijskom programu formalnih jezika i teorije automata, Chesnevar, González and Maguitman (2004) su ustanovili da se većina studenata ne oseća motivisanim i zainteresiranim za savladavanje ovih tema. Oni smatraju da je razlog nedostataka entuzijazma, ne samo složenost tema koje se obrađuju, već i činjenica da su teme bliže polju matematike nego računarstva. Zbog toga, u cilju unapređenja tradicionalne nastave predlažu nekoliko didaktičkih strategija, koje uključuju korišćenje simulatora kao nastavnih pomagala. Naveed and Sarim (2018), takođe, u svom radu predlažu didaktičku strategiju za učenje teorije automata i formalnih jezika, koja se sastoji od pet principa, od kojih se jedan odnosi na primenu softverskih alata u nastavi. Strategija je testirana na maloj grupi studenata i rezultati su bili prilično ohrabrujući.

Edukativni simulatori koji se koriste u oblasti programskih prevodioca su u stvari kompjuterski simulatori teorije koji studentima omogućavaju da „ožive“ mnoge apstraktne teme (Chesnevar, Cobo and Yurcik, 2003). Autori

u ovom radu analiziraju odabrane simulatore svrstavajući ih u dve osnovne grupe: generički, višenamenski softverski paketi koji integrišu nekoliko srodnih koncepata teorije automata i formalnih jezika i softverski alati dizajnirani za simulaciju određene klase automata. Višenamenski simulatori predstavljaju dobar izbor, jer pružaju isto okruženje za različite vrste automata, ali, takođe, njihovo istraživanje pokazuje da su mnogi studenti bili prilično zainteresovani da isprobavaju različite simulatore za isti automat, dok su očekivanja bila da će se fokusirati na jedan simulator.

Ideja o upotrebi simulacionih alata u nastavi nije nova, u (Chakraborty, Saxena and Katti, 2011) se daje pregled simulacionih alata koji obuhvata vremenski period od 50 godina. Međutim, evidentno je da se poslednjih nekoliko godina značajnije radi na razvoju sofisticiranijih simulacionih softvera za edukaciju programskih prevodioca. Postoji veliki broj takvih simulatora, od jednostavnih sa prilično oskudnim funkcijama do višenamenskih koji nude moćne alate za interaktivnu edukaciju.

U radu (Boyd and Whalley, 1993), autori su nastojali da, studentima približe proces optimizacije međukoda. Razvili su alat *xvpodb* koji studentima omogućuje interaktivnu vizualizaciju efekta svake transformacije međukoda, kako bi se izvršila veoma korisna ilustracija procesa optimizacije. Ovaj alat za vizualizaciju razvijen je da podrži analizu i pregled optimizacija koje vrši *vpo* optimizator (Benitez and Davidson, 1988). Optimizator može iterativno da primenjuje faze optimizacije više puta. Alat *xvpodb* prepoznaje ne samo promene koje su nastale, već i kada, odnosno u kojoj fazi su nastale tokom procesa optimizacije.

LLparse i LRparse (Blythe, James and Rodger, 1994) su dva interaktivna edukativna alata za vizuelizaciju procesa LL i LR parsiranja. Ovi alati se mogu koristiti za objašnjenje postupka generisanja LL(1) i LR(1) sintaksnih tabela kroz niz koraka, pri čemu, korisnici dobijaju povratnu informaciju o ispravnosti svakog koraka pre prelaska na sledeći. Na primer, kod LRparse, korisnik inicijalno unosi LR(1) gramatiku, izračunava FIRST i FOLLOW skupove, grafički konstruiše deterministički konačni automat i nakon toga formira LR(1) sintaksnu tabelu. Po završetku svih ovih koraka, korisnik može posmatrati vizualizaciju parsiranja ulaznih nizova.

Visual YACC (White, Ruby and Deddens, 1999) je još jedan alat za praćenje procesa parsiranja. U pitanju je proširenje za YACC parser, koje obezbeđuje vizuelni prikaz LR parsiranja za zadatu gramatiku i ulazni niz. Korisnici ovog alata mogu da prate postupak analize ulaznog niza, kao i proces formiranja sintaksnog stabla. Verzija ovog softvera koja je predviđena za studente uključuje desetak osnovnih primera, tako da učenje može da započne posmatranjem ovih primera, menjanjem zadatih gramatika i ulaznih nizova.

The Java Computability Toolkit – JCT (Robinson et al., 1999) 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.

Language emulator (Vieira et al., 2004) je softver razvijen u Javi koji omogućava manipulaciju regularnim izrazima, regularnom gramatikom, determinističkim konačnim automatama, nedeterminističkim konačnim automatama, nedeterminističkim konačnim automatama sa epsilon prelazima, Murovom i Milijevom mašinom. Studenti mogu na primer, da kreiraju automate, definišu regularne gramatike, da izračunavaju unije i preseke defiisanih gramatika, da vrše konverziju nedeterminističkog u deterministički konačni automat.

Edukativni sistem LISA (Language Implementation System Based on Attribute Grammars) pruža mogućnost eksperimentiranja i testiranja različitih leksičkih i sintaksnih analizatora (Mernik and Zumer, 2003). LISA je integrisano razvojno okruženje u kojem korisnici mogu da konstruišu prednji deo programskog prevodioca (koji obuhvata faze analize) i testiraju novoformirani jezik. Studenti mogu pratiti proces programskog prevođenja i steći intuitivno razumevanje leksičke, sintaksne i semantičke faze, kroz odgovarajuće animacije. U leksičkom delu alata studenti uče regularne izraze, konačne automate i upoznaju se sa različitim mogućnostima njihove primene. U sintaksnom delu alata uče LL(k) i LR(k) parsere. Na kraju, u semantičkom delu alata, studenti se upoznaju sa atributnim gramatikama.

Autori u radu (Castro-Schez et al., 2009) predstavljaju SoftwarE for Learning Formal languages and Automata theory (SELFA), simulacioni edukativni alat osmišljen sa ciljem da unapredi kvalitet nastave. U pitanju je web aplikacija koja razlikuje dve vrste korisnika - studenti i profesori. Profesorima je omogućen pregled aktivnosti studenata. SELFA omogućava snimanje i praćenje rezultata rada svakog studenta. Ove informacije se koriste za evaluaciju rada učenika. Studenti mogu da eksperimentišu sa gramatikom (regularnom ili beskonteksnom), kao i sa automata (konačni ili push-down).

Marcial-Romero et al. (2009) opisuju edukativni softver FlyA, koji su razvili u skladu sa odgovarajućim pedagoškim metodologijama. FLyA se sastoji od tri modula: modul za konačne automate, modul za beskonteksne gramatike i modul za Tjuringovu mašinu. U ovom radu je najviše prostora posvećeno opisu pedagoških metodologija koje su korišćene za razvoj softvera.

Interaktivni edukativni softver RegExpert (Budiselic, Srbljic and Popovic, 2007) predstavlja alat za manipulaciju regularnim jezicima. Glavni cilj alata RegExpert je da pojednostavi i vizuelno predstavi kompleksne koncepte teorije automata i formalnih jezika. RegExpert konvertuje korisnički definisani ili automatski generisani regularni izraz u ekvivalentni nedeterministički konačni automat sa epsilon prelazima i predstavlja ga korisniku u obliku dijagrama stanja. Alat omogućava studentima da eksperimentišu sa različitim regularnim izrazima i vide kako promene utiču na rezultujući automat.

U radu (Chakraborty, Saxena and Katti, 2013) predstavljen je jezik FADL (Finite Automaton Description Language) i simulacioni alat koji je zasnovan na tom jeziku. FADL je u tom radu prikazan kao jednostavan jezik koji se može koristiti za definisanje determinističkih i nedeterminističkih konačnih automata. Upotreba ovog jezika ne zahteva nikakvu veštinu programiranja niti detaljno proučavanje specifikacija jezika. Alat sadrži šest međusobno povezanih programa i to FADL programski prevodilac (FADLC), FADL optimizator (FADLOC), konačni automat interpretator (FAI), vizuelizator dijagrama prelaza (TDV), konvertor nedeterminističkog u deterministički konačni automat (NFADFAT) i konvertor determinističkog konačnog automata u Tjuringovu mašinu (DFATMT). Svih šest programa realizovano je na programskom jeziku C ++. Alat je baziran na konzoli i svaki pojedinačni alat se poziva kratkom naredbom u komandnoj liniji.

Autori u (Kundra and Sureka, 2016) predlažu edukativno okruženje CPLC (Case-based and Project-based Learning environment for teaching Compiler design) za unapređenje učenja programskih prevodioca. CPLC se konkretno primenjuje za fazu analize kod programskih prevodioca – leksičku i sintaksnu. Ovaj model koristi slučajeve koji se zasnivaju na praktičnim problemima pružajući studentima iskustvo u rešavanju složenih problema iz realnog sveta. Model na kraju predviđa i mini projekat, koji doprinosi boljem razumevanju procesa programskog prevođenja.

PAVT (Parsing Algorithm Visualizer Tool) je simulacioni alat za vizuelizaciju postupka konstrukcije parsera, za zadatu beskonteksnu gramatiku (Sangal et al., 2018). Nakon konstrukcije parsera ilustruje se proces analize za određeni ulazni niz. PAVT podržava šest različitih algoritama za parsiranje, prezentujući svaki korak odabrane analize, FIRST i FOLLOW skupove, sintaksne tabele, sintaksno stablo.

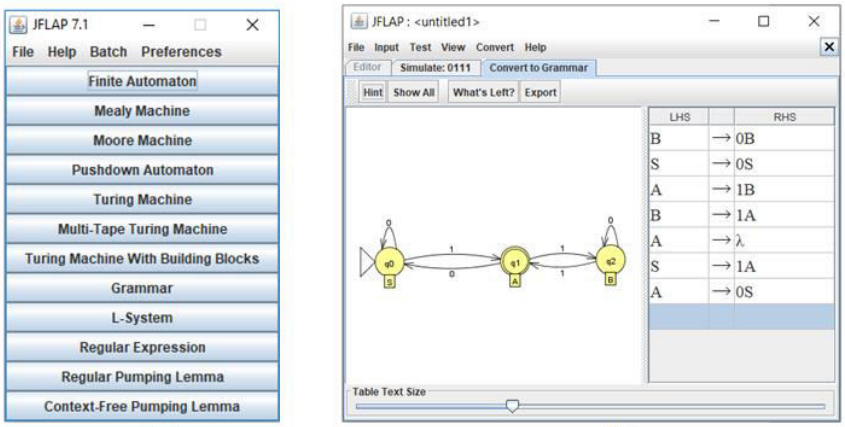
Navedeni softverski alati predstavljaju samo deo simulacionih sistema koji su razvijeni za potrebe edukacije programskih prevodioca. Ovde su ukratko predstavljeni samo kao reprezentativni primeri, kako bi se stekao uvid u raznolikost funkcionalnosti i mogućnosti koje nude. Svaki od ovih simulatora

je specifičan na svoj način, obrađuje jednu ili više tema, ima odgovarajući pristup rešavanju problema i koristi određenu didaktičku metodu za uspostavljanje interakcije sa studentom. U narednom poglavlju su detaljnije analizirana četiri odabrana simulaciona sistema: JFLAP kao jedan od najpopularnijih i najkompletnijih alata, jFAST kao jednostavan i praktičan alat za simulaciju konačnih automata, i dva novija rešenja za učenje formalnih jezika i teorije automata, Seshat web bazirana aplikacija i Automata Simulator mobilna android aplikacija.

5. ANALIZA ODABRANIH SIMULACIONIH SISTEMA

5.1. JFLAP

JFLAP (Java Formal Languages and Automata Package) je interaktivni grafički softverski alat, namenjen za edukaciju formalnih jezika, teorije automata i programskih prevodioca (Procopiuc, Procopiuc and Rodger, 1996; Rodger and Finley, 2006). Razvoj JFLAP-a započeo je 1990. godine najpre u vidu jednostavnije verzije FLAP (LoSacco and Rodger, 1993) napisane u C++ i X windows, da bi kasnije FLAP bio prekodiran u Javi, i tako postao JFLAP. Za razvoj ovog alata zaslužna je profesorka Susan Rodger i njena istraživačka grupa na Univerzitetu Duke u Severnoj Karolini. JFLAP je višenamenski softverski paket koji nudi širok spektar mogućnosti, od eksperimentisanja sa regularnim izrazima i gramatikama do konstrukcije teorijskih mašina i automata, što se može videti na slici 1a.

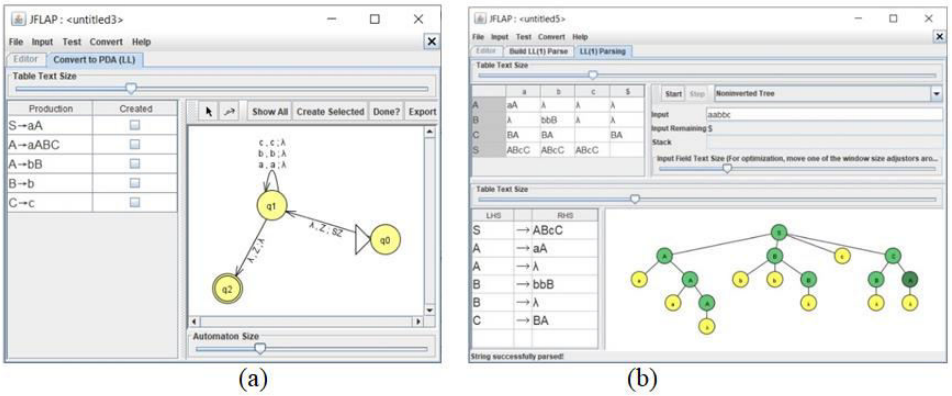


Slika 1. *Opcije koje nudi JFLAP prilikom pokretanja (a) i konverzija determinističkog konačnog automata u regularnu gramatiku (b)*

Pored osnovnih funkcionalnosti koje se odnose na definisanje automata i gramatike, JFLAP pruža mogućnost konstruisanja forme, putem transformacija iz jednog oblika u drugi (Rodger, Lim and Reading, 2007). Na primer, nakon konstrukcije nedeterminističkog konačnog automata, može se

izvršiti konverzija u deterministički konačni automat, zatim se može izvršiti minimizacija i na kraju transformacija u regularnu gramatiku (slika 1b).

Takođe, moguće je na osnovu definisane beskonteksne gramatike konstruisati, ekvivalentni nedeterministički push-down automat (slika 2a). JFLAP korisnicima omogućuje i eksperimentisanje sa teorijskim materijalom. Na primer, upoznavanje sa procesom parsiranja, vrši se tako što se najpre formira sintaksna tabela, a zatim se analizira ulazni niz, korak po korak, formirajući ekvivalentno sintaksno stablo (slika 2b). Konstrukcija sintaksne tabele uključuje formiranje posebnog determinističkog konačnog automata koji vrši proces parsiranja. Na ovaj način predstavljena je praktična upotreba determinističkog konačnog automata.



(a) (b)

Slika 2. *Konverzija beskonteksne gramatike u nedeterministički push-down automat (a) i proces parsiranja za zadati ulazni niz uz kreiranje sintaksne tabele i sintaksnog stabla (b)*

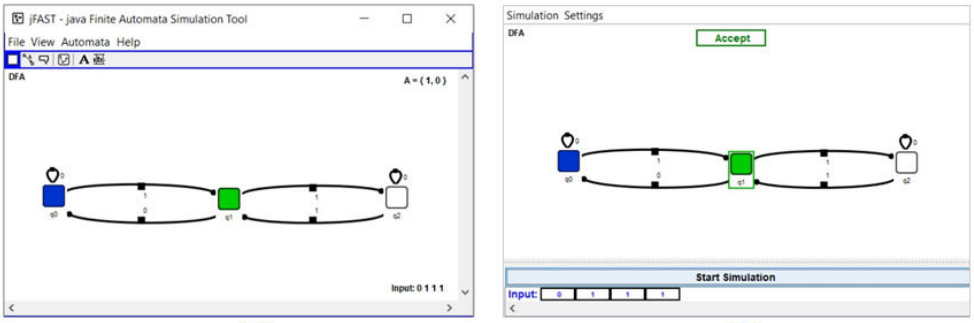
Na osnovu analize ovog simulacionog sistema može se zaključiti da se radi intuitivnom softveru, veoma jednostavanom za korišćenje koji se može besplatno preuzeti na sajtu Univerziteta Duke. Graf automata se veoma jednostavno konstruiše na radnoj površini, tako što se stanja automata dodaju izborom odgovarajućeg alata i levim klikom, dok se prelazi dodaju prevlačenjem miša od jednog do drugog stanja. Mnoge automatizovane opcije kao što su konverzije iz jedne u drugu formu, formiranje sintaksne tabele i sintaksnog stabla, mogu se izvršavati u koracima, tako da korisnici dobijaju precizne povratne informacije o svakom procesu koji se izvrašava. Ovaj edukativni simulator koriste brojni fakulteti u svom nastavnom programu (Rodger et al., 2009), upravo zbog toga što je jedan od najpotpunijih alata iz ove oblasti, velikog broja funkcionalnosti koje pruža i zbog jenostavnosti korišćenja. U pitanju je java desktop aplikacija koja se jednostavno pokreće na desktop računarima. Kao nedostatak može da se napomene da ovaj softver nije kompatibilan sa prenosnim uređajima kao što su tablet i telefon, a koji su izuzetno popularni kod studenata. Za potrebe

analize u ovom radu, korišćena je poslednja stabilna verzija ovog softvera 7.1. Verzija 8.0 objavljena je kao beta verzija još januara 2015. godine.
Autori u radu (Yalagi and Dixit, 2017) izveli su eksperiment sa oko 70 studenata, kojima su pored klasične teorijske nastave uvedene i laboratorijske vežbe sa simulacionim softverom JFLAP. Postignute rezultate studenata na kraju godine upoređivali su sa rezultatima studenata iz prethodne akademske godine u kojoj nije korišćen JFLAP. Uočeno je da su rezultati na završnom ispitu bolji za više od 20%.

5.2. jFAST

jFAST (Java Finite Automata Simulation Tool) je robusna i intuitivna aplikacija, implementirana u Javi, koja korisniku pruža mogućnost da na brz i jednostavan način konstruiše i modifikuje automate (White and Way, 2006). Ovaj edukativni simulator za proučavanje automata, dizajniran je da bude alternativa naprednijem JFLAP-u. Kreiranje automata je, kao i kod FLAP-a, veoma jednostavno u okviru intuitivnog grafičkog korisničkog interfejsa. Jednim klikom dodaje se stanje automata, a prevlačenjem crtaju se odgovarajući prelazi. Dodatna svojstva određenog stanja mogu se definisati preko dijalog boksa, koji se dobija desnim klikom na željeni element. jFAST pruža modularni pristup za formiranje složenih automata primenom tako zvanih pod-automata. Automati se mogu kreirati iz više delova, ili 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.

Primer konstrukcije jednog determinističkog konačnog automata prikazan je na slici 3a. Može se primetiti da je grafički prikaz sličan kao kod JFLAP-a. Takođe, i kod jFAST-a se konstruisani automat može sačuvati, i po potrebi ponovo učitati.



(a) (b)

Slika 3. *Primer konstrukcije determinističkog konačnog automata (a) i jFAST simulacioni interfejs (b)*

Izborom odgovarajuće opcije unosi se ulazni niz i nakon toga se može pokrenuti simulacija. Simulacioni interfejs koristi se za prikaz postupne simulacije rada automata za dati ulaz (slika 3b).

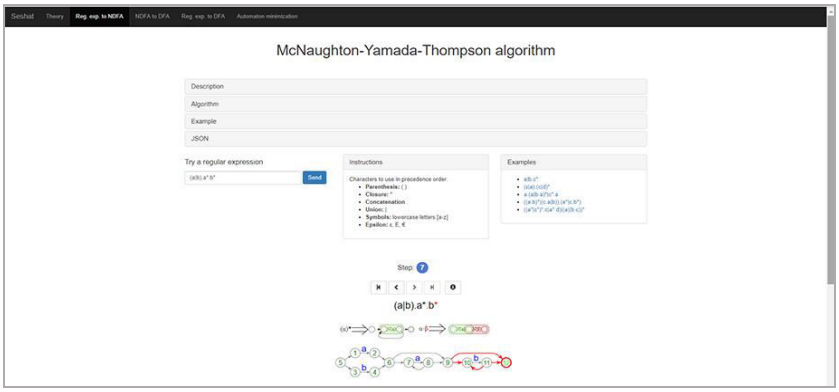
Na osnovu ove analize može se ustanoviti da jFAST softver obezbeđuje jednostavnu platformu za učenje teorije automata kroz interaktivni dizajn i grafičku simulaciju. U pitanju je moćan alat, ali za razliku od JFLAP-a pruža daleko manji broj opcija i funkcionalnosti. Radno okruženje za sve podržane tipove automata je isto. U pitanju je Java desktop aplikacija koja se jednostavno instalira, u par koraka. Takođe, nije kompatibilana sa mobilnim uređajima. Za testiranje je korišćena verzija softvera 1.2 iz 2006. godine. Novije verzije se ne mogu naći, što znači da se ovaj softver više ne razvija.

Autori ovog softvera predstavili su njegovu evaluaciju od strane 18 studenata. Studenti su nakon upotrebe jFAST-a u nastavi, popunili upitnik koji se odnosio na njihovo iskustvo pre i posle korišćenja ovog simulatora. Od 18 studenata, 16 je smatralo da je jFAST veoma jednostavan za upotrebu, 15 studenata da je njegov interfejs intuitivan, a svih 18 je smatralo da korištenje jFAST-a u kombinaciji sa teorijom značajno unapređuje njihovo razumevanje teorije automata.

5.3. SESHAT

Seshat je edukativni alat, baziran na web tehnologiji, namenjen za učenje osnovnih koncepata regularnih jezika, regularnih izraza i konačnih automata (Arnaiz-González et al., 2018). Alat, nazvan po egipatskoj boginji mudrosti i znanja, pruža studentima mogućnost interakcije i eksperimentiranja sa najčešćim algoritamima leksičke analize i teorije automata. Ova web aplikacija je dizajnirana tako da se web elementi prilagođavaju okruženju, što garantuje dobru preglednost na različitim uređajima. Kada se Seshat pokrene inicijalna web stranica aplikacije prikazuje sve implementirane algoritme sa kratkim opisom. 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.

Prva tri algoritma rade sa regularnim izrazima, dok poslednji radi sa automatama, zbog čega je interfejs web stranice tog algoritma drugačiji. Algoritmi koji rade sa regularnim izrazima imaju ista korisnička okruženja koja se sastoje iz dva dela: statički deo u kojem se daje teorijsko objašnjenje funkcionisanja algoritma i dinamički deo, u kojem studenti prate kako algoritam funkcioniše (slika 4). Studenti mogu da izaberu ponuđene primere regularnih izraza ili da kreiraju svoje.



Slika 4. *Formiranje nedeterminističkog konačnog automata za zadati regularni izraz*

Algoritam se može izvršavati korak po korak, a postoji i opcija vraćanja unazad, ali i mogućnost da se direktno ode do poslednjeg koraka. U svakom koraku se ističu aktuelni delovi automata, sintaksnog stabla ili tabela koje generišu algoritmi (slika 5).

Σ	p	followpos(p)
a	1	4
b	2	3 4
c	3	3 4

Q/Σ	a	b	c	Set of positions
A	B	C	D	1 2
(B)	D	D	D	4
(C)	D	D	C	3 4
D	D	D	D	

Slika 5. *Formiranje tranzicione tabele prilikom konstrukcije determinističkog konačnog automata za zadati regularni izraz*

Za funkciju minimizacije automata potreban je kao ulaz automat čiji se broj stanja treba smanjiti. Interfejs ove funkcije je zbog toga drugačiji, i sastoji se od polja za crtanje stanja i prelaza automata. Seshat je klijent server aplikacija, gde je klijent dinamička web stranica koja u interakciji sa korisnikom šalje zahteve prema serveru. Server je napisan na Python-u, a za razvoj korisničkog interfejsa korišćen je HTML, SVG, JavaScript, jQueri i CSS. Lekser i parser za regularne izraze su izrađeni pomoću PLY (Python-ova implementacija Lex-a i Yacc-a).

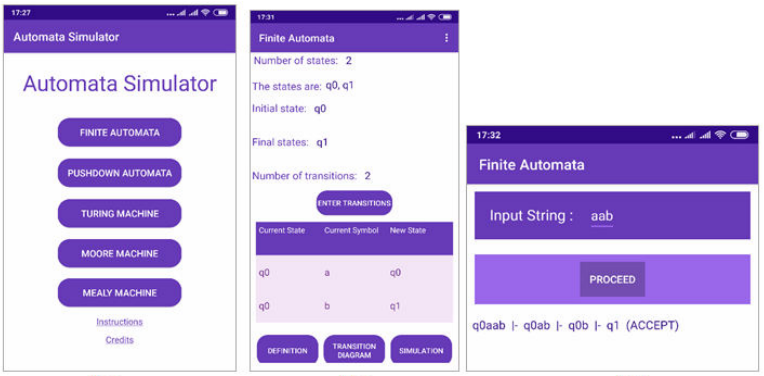
Analizom je utvrđeno da ovaj edukativni alat nudi ograničene mogućnosti u pogledu rada sa automatima. Automate je moguće kreirati samo unosom odgovarajućeg regularnog izraza. Dakle, nije moguće kreirati automat na osnovu gramatike i nije moguće testirati rad konstruisanog automata za neki zadati ulazni niz. Takođe, nije moguća obrnuta funkcija, da se na osnovu kreiranog automata generiše odgovarajuća gramatika. Kao dobru stvar treba

istaći dostupnost teorije u statičkom delu stranice, kao i prezentovanje kratkih informacija, u svakom koraku simulacije, o pravilima koja se primenjuju u tom trenutku. Analizom je utvrđeno i da ovaj alat ne nudi mogućnost čuvanja i ponovnog pokretanja kreiranog automata. Ono što nudi, je eksporovanje konačnog rešenja algoritma u JSON formatu, i to ne direktnim odabirom odgovarajuće opcije, već unosom posebnog URI (Uniform Resource Identifier) za svaki algoritam u polju za pretragu web pretraživača. Ovaj web simulator iako ima mnogo manje funkcija od JFLAP-a predstavlja odličnu osnovu za dalji razvoj novih i unapređivanje postojećih. Pomenuta web tehnologija ga čini pristupačnijim od ostalih alata, jer se može pokrenuti u okviru bilo kog operativnog sistema i na bilo kom uređaju, potreban je samo web pretraživač.

Autori ovog alata izvršili su eksperiment u kome je učestvovalo 56 studenata, tako što su bili podeljeni u dve grupe, i to jedna koja je samo pratila teorijska predavanja i druga koja je pored predavanja koristila i Seshat. Studentima je usledio test, čiji su rezultati potvrdili da je upotreba ovog alata imala pozitivne efekte. Najveću ocenu su u većem procentu dobili studenti iz grupe koja je koristila Seshat.

5.4. AUTOMATA SIMULATOR

Automata Simulator je mobilna aplikacija koja obezbeđuje edukativno okruženje za simulaciju više tipova automata (Singh et al., 2019). Ova aplikacija, razvijena za mobilne telefone i tablete sa Android operativnim sistemom, 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, Tjuringovu mašinu, kao i Murovu i Milijevu mašinu (slika 6a).



Slika 6. *Početni ekran aplikacije Automata Simulator (a), definisanje konačnog automata u aplikaciji Automata Simulator (b) i simulacija konstruisanog konačnog automata (c)*

Kada se izabere tip automata otvara se forma za definisanje osnovnih parametara izabranog automata, kao i tabele prelaza (slika 6b). Nakon unosa svih neophodnih podataka, može se izabrati jedna od tri ponuđenih opcija: može se pogledati konačna definicija automata, graf automata ili pokrenuti simulacija. Simulacija konstruisanog automata podrazumeva unos ulaznog niza, nakon čega se ispisuju koraci analize datog niza (slika 6c).

Kao što se iz navedenog može videti, Automata Simulator ima vrlo skromne mogućnosti u odnosu na prethodna tri analizirana simulaciona alata, ali ipak zavređuje pažnju zbog toga što se radi o mobilnoj aplikaciji. Istraživanja govore da je, krajem 2016. godine, upotreba pametnih mobilnih telefona i tablet uređaja premašila upotrebu desktop računara. Prenosni uređaji su naročito popularni kod mlađih generacija, zbog toga treba raditi na razvoju edukativnih aplikacija za ove uređaje.

Autori ove edukativne simulacione aplikacije sproveli su istraživanje na kraju semestra, u kojem se pored uobičajene nastave koristio ovaj alat kao i JFLAP zbog naprednijih koncepata iz oblasti teorije automata. Test je pokazao da je ovakav pristup pomogao studentima da bolje savladaju teoriju, a naročito teme koje se odnose na projektovanje automata. Takođe, studenti su smatrali da kombinacija ova dva simulaciona alata čine nastavu interaktivnijom što pozitivno utiče na proces učenja.

6. UPOREDNE KARAKTERISTIKE ANALIZIRANIH SIMULATORA

Na osnovu izvršene analize odabranih simulatora mogu se sumirati rezultati i prikazati uporedne karakteristike. U tabeli 1. prikazane su tehničke karakteristike simulacionih sistema.

Tabela 1. Tehničke karakteristike analiziranih simulacionih sistema

Simulator	*Programski jezik*	*Platforma*	*Potrebna instalacija*	*Dostupnost*
JFLAP	Java	Desktop	Da	Besplatno
jFAST	Java	Desktop	Da	Besplatno
Seshat	Python, HTML, SVG, JavaScript	Web	Ne	Besplatno
Automata Simulator	Java	Mobilna	Da	Besplatno

Uporedni prikaz tema, iz oblasti programskih prevodioca, koje obrađuju navedeni simulatori dat je u tabeli 2.

Tabela 2. Uporedni prikaz funkcija analiziranih simulatora

Funkcije	*JFLAP*	*jFAST*	*Seshat*	*Automata Simulator*
Opis jezika	da	ne	da	ne
Konstruisanje automata	da	da	da	da
Konverzija automata	da	ne	da	ne
Prepoznavanje jezika	da	da	ne	da
Preslikavanje regularnih izraza u automate	da	ne	da	ne
Leksička analiza	da	ne	delimično	ne
Sintaksna analiza	da	ne	delimično	ne
Generisanje međukoda	ne	ne	ne	ne
Generisanje koda	ne	ne	ne	ne

6. ZAKLJUČAK

U ovom radu opisan je veći broj simulacionih sistema koji obrađuju različite teme iz oblasti programskih prevodioca. Veliki broj teorijskih apstraktnih koncepata iz različitih faza programskog prevođenja može se ilustrovati pomoću takvih simulatora. Na osnovu izvršene analize može se zaključiti da simulacioni sistemi predstavljaju odličan spoj teorijskog i praktičnog iskustva, i da kao takvi doprinose unapređenju nastavnog procesa. Rezultati brojnih istraživanja ukazuju na pozitivne efekte korišćenja ovih edukativnih alata, naročito ukoliko se koriste kao pomoćno nastavno sredstvo u kombinaciji sa tradicionalnom nastavom. Najviše su zastupljeni edukativni sistemi koji simuliraju konačne automate, kao i sistemi za simulaciju leksičke i sintaksne analize.

Od analiziranih alata kao najbolji u pogledu funkcionalnosti, grafičkog prikaza i jednostavnosti korišćenja izdvaja se JFLAP. U pitanju je višenamenski simulacioni alat, koji vizualizuje veliki broj teorijskih koncepata iz oblasti programskih prevodioca. Jedini nedostatak JFLAP-a, kao desktop aplikacije, je problem kompatibilnosti na različitim uređajima. U ovom radu analiziran je i web alat Seshat, koji uprkos svojim nedostacima predstavlja dobru osnovu za dalje usavršavanje. Dizajn edukativnih simulacionih softvera treba usmeriti prema web tehnologijama, jer se time rešavaju problemi instalacije i kompatibilnosti sa različitim platformama. Preporuka za dalji razvoj simulacionih web alata je da se u okviru tih aplikacija predvidi i mogućnost čuvanja podataka o sprovedenim procesima simulacije, kako bi student mogao da nastavi sa radom ukoliko nije završio proces simulacije. Zbog toga treba raditi na razvoj sistema za prijavu sa različitim nivoima pristupa (student, profesor), koji bi omogućio i čuvanje

podataka na cloud nalogu korisnika simulacione web aplikacije. Na taj način bi, na bilo kom uređaju i u bilo kom trenutku, prijavljeni korisnik imao pristup svom edukativnom okruženju i svojim rezultatima rada.

Iako je veliki broj simulacionih sistema danas dostupan, potrebno je i dalje razvijati nove ili usavršavati postojeće, pre svega, zato što treba ići u korak sa novim tehnologijama, kao i zbog toga što svaki naredni alat dolazi sa nekim novim doprinosom u pogledu usavršavanja nastavnog procesa.

REFERENCE

1. Adar, E., 2006. Guess: A language and interface for graph exploration. *Proceedings of the 2006 Conference on Human Factors in Computing Systems*, pp. 347–363.
2. Aho, A. V., Lam, M. S., Sethi, R. and Ullman, J. D., 2007. *Compilers: principles, techniques and tools*, 2nd ed. Boston: Pearson Education Inc.
3. Arnaiz-González, Á., Díez-Pastor, J. F., Ramos-Pérez, I. and García-Osorio, C., 2018. Seshat—a web-based educational resource for teaching the most common algorithms of lexical analysis. *Computer Applications in Engineering Education*, 26(6), 2255-2265.
4. Benitez, M. E., and Davidson, J. W., 1988. A portable global optimizer and linker. *Proceedings of the SIGPLAN '88 Symposium on Programming Language Design and Implementation*, 23(7), pp. 329-338.
5. Blythe, S. A., James, M. C. and Rodger, S. H., 1994. LLparse and LRparse: visual and interactive tools for parsing. *ACM SIGCSE Bulletin*, 26(1), pp.208-212.
6. Boyd, M. and Whalley, D., 1993. Graphical visualization of compiler optimizations. *Journal of Programming Languages*, 3(2), pp. 69-94.
7. Budiselic, I., Srbljic, S., and Popovic, M., 2007. Regexpert: A tool for visualization of regular expressions. *In EUROCON 2007-The International Conference on" Computer as a Tool"*, pp. 2387-2389.
8. Castro-Schez, J. J., Del Castillo, E., Hortolano, J. and Rodriguez, A., 2009. Designing and using software tools for educational purposes: FLAT, a case study. *IEEE Transactions on Education*, 52(1), pp. 66-74.
9. Chakraborty, P., Saxena, P. C. and Katti, C. P., 2013. A compiler-based toolkit to teach and learn finite automata. *Computer Applications in Engineering Education*, 21(3), pp. 467-474.
10. Chakraborty, P., Saxena, P. C., and Katti, C. P., 2011. Fifty years of automata simulation: a review. *ACM Inroads*, 2(4), pp. 59-70.

11. Chesnevar, C. I., Cobo, M. L., and Yurcik, W., 2003. Using theoretical computer simulators for formal languages and automata theory. *ACM SIGCSE Bulletin*, 35(2), pp. 33-37.
12. Chesnevar, C. I., González, M. P. and Maguitman, A. G., 2004. Didactic strategies for promoting significant learning in formal languages and automata theory. *ACM SIGCSE Bulletin*, 36(3), pp. 7-11.
13. Djordjevic, J., Nikolic, B. and Milenkovic, A., 2005. FlexibleWeb-Based Educational System for Teaching Computer Architecture and Organization. *IEEE Transactions on Education*, 48(2), pp. 264-273.
14. Fitch, W. T. and Friederici, A. D., 2012. Artificial grammar learning meets formal language theory: an overview. *Philosophical Transactions of the Royal Society B*, 367(1598), pp. 1933-1955.
15. Huppert, J., Lomask, S. M. and Lazarowitz, R., 2002. Computer simulations in the high school: Students' cognitive stages, science process skills and academic achievement in microbiology. *International Journal of Science Education.* 24(8), pp. 803-821.
16. Indu, J., 2016. Technique for Conversion of Regular Expression to and from Finite Automata. *International Journal of Recent Research Aspects*, Vol. 3(2), pp 62-64.
17. Jäger, G., and Rogers, J., 2012. Formal language theory: refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B*, 367(1598), pp. 1956-1970.
18. Jovanović, N. and Zakić, A., 2018. Network simulation tools and spectral graph theory in teaching computer network. *Computer Applications in Engineering Education*, 26(6), pp. 2084-2091.
19. Jovanović, N., Jovanović, Z. and Jevremović, A., 2016. Evaluation of simulators for teaching computer networks. *The International journal of engineering education*, 32(5), pp. 2098-2106.
20. Jovanović, N., Popović, R. and Jovanović, Z., 2009. WNetSim: a web-based computer network simulator. *International Journal of Electrical Engineering Education*, 46(4), pp. 383-396.
21. Jovanović, N., Popović, R., Marković, S. and Jovanovic, Z., 2012. Web laboratory for computer network. *Computer Applications in Engineering Education*, 20(3), pp. 493-502.
22. Jovanović, N., Zakić, A., and Veinović, M., 2016. VirtualMeshLab: Virtual laboratory for teaching wireless mesh network. *Computer Applications in Engineering Education*, 24(4), pp. 567-576.
23. Knobelsdorf, M., Kreitz, C. and Böhne, S., 2014. Teaching theoretical computer science using a cognitive apprenticeship approach. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pp. 67-72.

24. Konold, C., and Miller, C. D., 2005. *TinkerPlots: Dynamic data exploration*. Computer software. Emeryville, CA: Key Curriculum Press.
25. Koparan, T., and Yilmaz, G. K., 2015. The Effect of Simulation-Based Learning on Prospective Teachers' Inference Skills in Teaching Probability. *Universal Journal of Educational Research*, 3(11), pp. 775-786.
26. Kundra, D. and Sureka, A., 2016. An experience report on teaching compiler design concepts using case-based and project-based learning approaches. *In 2016 IEEE Eighth International Conference on Technology for Education (T4E)*, pp. 216-219.
27. Lindgren, R., and Schwartz, D., 2009. Spatial Learning and computer simulations in science. *International Journal of Science Education*, 31(3), pp. 419–438.
28. Linz, P., 2012. *An introduction to formal languages and automata,* 5th ed. London: Jones & Bartlett Learning.
29. LoSacco, M. and Rodger, S., 1993. FLAP: A tool for drawing and simulating automata. *EDMEDIA 93*, pp. 310-317.
30. Marcial-Romero, J. R., Alvarez Contreras, P. A., Montes Venegas, H. A., and Hernández Servín, J. A., 2009. A simulator for teaching automatas and formal languages: FLyA. *ICEIS 2009 - 11th International Conference on Enterprise Information Systems, Proceedings*, pp. 175–178.
31. Mernik, M. and Zumer, V., 2003. An educational tool for teaching compiler construction. *IEEE Transactions on Education*, 46(1), pp. 61-68.
32. Nahvi, M., 1996. Dynamics of student-computer interaction in a simulation environment: Reflections on curricular issues. *Proceedings of the IEEE Frontiers in Education, USA*, pp. 1383-1386.
33. Naveed, M. S. and Sarim, M., 2018. Didactic Strategy for Learning Theory of Automata & Formal Languages. *Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences*, 55(2), pp. 55-67.
34. Procopiuc, M., Procopiuc, O. and Rodger, S. H., 1996. Visualization and interaction in the computer science formal languages course with JFLAP. *In Frontiers in Education FIE'96 26th Annual Conference*, pp. 121-125.
35. Robinson, M. B., Hamshar, J. A., Novillo, J. E. and Duchowski, A. T., 1999. A Java-based tool for reasoning about models of computation through simulating finite automata and Turing machines. *In ACM SIGCSE Bulletin*, 31(1), pp. 105-109).
36. Rodger, S. H. and Finley, T. W., 2006. *JFLAP: an interactive formal languages and automata package*. London: Jones & Bartlett Publishers.

37. Rodger, S. H., Lim, J. and Reading, S., 2007. Increasing interaction and support in the formal languages and automata theory course. *In ACM SIGCSE Bulletin*, 39(3), pp. 58-62.
38. Rodger, S. H., Wiebe, E., Lee, K. M., Morgan, C., Omar, K. and Su, J., 2009. Increasing engagement in automata theory with JFLAP. In *ACM SIGCSE Bulletin*, 41(1), pp. 403-407.
39. Rutten, N., Van Joolingen, W. R., and Van der Veen, J. T., 2012. The learning effects of computer simulations in science education. *Computers & Education, Elsevier*, 58(2012), pp. 136-153.
40. Sangal, S., Kataria, S., Tyagi, T., Gupta, N., Kirtani, Y., Agrawal, S. and Chakraborty, P., 2018. PAVT: a tool to visualize and teach parsing algorithms. *Education and Information Technologies*, 23(6), pp. 2737-2764.
41. Singh, T., Afreen, S., Chakraborty, P., Raj, R., Yadav, S. and Jain, D., 2019. Automata Simulator: A mobile app to teach theory of computation. Computer Applications in Engineering Education, [online]. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.22135 [Accessed 11 Aug. 2019].
42. Soderberg, P., and Price, F., 2003. An examination of problem-based teaching and learning in population genetics and evolution using EVOLVE, a computer simulation. *International Journal of Science Education*, 25(1), pp. 35-55.
43. Stanković, M., Stojković, S. and Tošić, Ž., 2018. *Programski prevodioci*. Niš: Elektronski fakultet.
44. Su, Y. and Yan, S. Y., 2012. *Principles of Compilers: A New Approach to Compilers Including the Algebraic Method*. Beijing: Higher Education Press.
45. Taher, M., and Khan, A., 2014. Impact of Simulation-based and Hands-on Teaching Methodologies on Students' Learning in an Engineering Technology Program. *Proceedings of the ASEE Annual Conference & Exposition* , pp. 1-22.
46. Vieira, L. F. M., Vieira, M. A. M. and Vieira, N. J., 2004. Language emulator, a helpful toolkit in the learning process of computer theory. *In ACM SIGCSE Bulletin*, 36(1), pp. 135-139.
47. White, E. L., Ruby, J. and Deddens, L. D., 1999. Software visualization of LR parsing and synthesized attribute evaluation. *Software: Practice and Experience*, 29(1), pp. 1-16.
48. White, T. M. and Way, T. P., 2006. jFAST: A java finite automata simulator. In *ACM SIGCSE Bulletin*, 38(1), 384-388.
49. Witzig, A., Prandini, M., Wolf, A., and Kunath, L., 2016. Teaching Renewable Energy Systems by Use of Simulation Software: Experience at Universities of Applied Sciences, in In-Service Training, and from International Know-How Transfer. *Presented at*

and in the proceedings of Eurosun.žWolfram, S., 2003. *The Mathematica Book*, 5th ed. Wolfram Media, Inc.

50. Yalagi, P. S. and Dixit, R. K., 2017. Enhancing the learning ability using JFLAP for Theory of Computation Course. *Journal of Engineering Education Transformations*, pp. 1-6.
51. Zhang, W. and Jie, L., 2018. Application of Simulation Software in Analog Electronic Technology Teaching. *DEStech Transactions on Social Science, Education and Human Science*, (ICESSH 2018), pp. 6-10.

Rad je primljen: 27.08.2019.
Prihvaćen za objavljivanje: 28.09.2019.
Received: 27 August, 2019
Accepted: 28 September, 2019