Obrazovni interaktivni alat za vizuelnu reprezentaciju leksičke analize

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

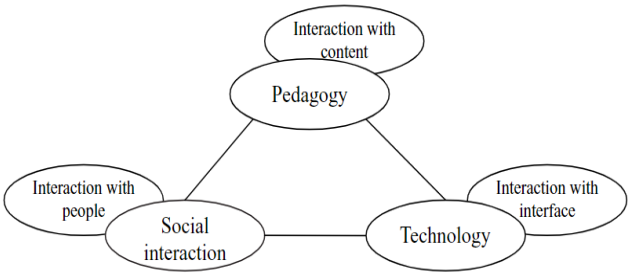
Sažetak—**U cilju unapređenja tradicionalne nastave, u mnogim oblastima računarskih nauka, uspešno se primenjuju alati za vizuelizaciju, simulatori, animacioni sistemi i drugi obrazovni softverski alati za ilustraciju i vizualizaciju apstraktnih koncepata. U ovom radu analizirani su takvi sistemi, posebno oni koji su pogodni za proučavanje prednjeg dela programskog prevodioca, koji obuhvata faze analize. Osnovni cilj ovog istraživačkog rada je razvoj interaktivnog softverskog sistema za vizuelizaciju leksičke analize u procesu programskog prevođenja. U pitanju je web bazirana aplikacija koja studentu pruža mogućnost da definiše regularnu gramatiku, unese određeni string i prati kako leksički analizator uzima niz karaktera, grupiše ih u lekseme i proizvodi izlaznu sekvencu tokena. Ovakav način interaktivnog eksperimentisanja sa teorijskim konceptima, treba da pojednostavi kompleksne teme i učini ih zanimljivijim.**

Ključne reči – leksička analiza; programski prevodioci; obrazovni alati; vizuelizacija algoritama;

I. UVOD

Interaktivnost je ključ efektivnog i efikasnog procesa podučavanja i učenja gde nastavnik može privući pažnju učenika, a učenici mogu naučiti više u poređenju sa tradicionalnom metodom. Termin „interaktivni“ pojavljuje se u dva različita sklopa obrazovnog istraživačkog diskursa: jedan koji se odnosi na pedagogiju, a drugi koji se tiče novih tehnologija u obrazovanju [1]. Pedagoška interaktivnost se odnosi na interakciju profesora i studenata. Upotreba tehnologije u obrazovanju podrazumeva efikasnu integraciju informaciono komunikacionih tehnologija (IKT) u nastavnom procesu. Wang u svom istraživanju [2] predlaže generički model za efikasnu integraciju IKT u procesu podučavanja i učenja. Po tom modelu, pedagogija, socijalna interakcija i tehnologija predstavljaju ključne komponente tehnološki unapređenog okruženja za učenje (Sl. 1).

U poslednjoj deceniji, ubrzanom progresijom računarskih kapaciteta i napretkom tehnologija grafičkog dizajna, multimedijalna okruženja za učenje evoluirala su od sekvencijalnog statičnog okvira za tekst i slike do visoko sofisticirane vizuelizacije [3]. IKT i posebno dizajnirani softverski proizvodi otvaraju nove mogućnosti u razvijanju potrebnih veština i sposobnosti studenata (praktične, verbalne, perceptivne, mentalne...) [4]. Može se reći da vizuelizacija pokreće maštu studenata i produbljuje njihovo razumevanje.



Slika 1. Interaktivnost na osnovu Wang modela [2].

Vizualizacija i interaktivnost predstavljaju nešto potpuno prirodno i očekivano, kada se govori o softveru koji se primenjuje u obrazovanju [5]. Dizajn obrazovnog softvera podrazumeva imaginaciju, ideju, elaboraciju i deskripciju računarskog sistema u odnosu na neke pedagoške ciljeve i različita obrazovna ograničenja koja treba uzeti u obzir u vezi sa okruženjem u kojem će se softver koristiti [6].

U ovom radu predstavljen je web bazirani softverski sistem koji je razvijen sa ciljem da studentima približi proces leksičke analize u procesu kompajliranja. Ovaj softver je zamišljen kao deo modularnog simulacionog sistema koji će biti realizovan tako da pokrije što veći broj tema iz oblasti programskog prevođenja. Rad je sistematizovan u četiri poglavlja. Drugo poglavlje prezentuje obrazovne softverske sisteme za vizuelnu reprezentaciju apstraktnih teorijskih koncepata, sa kratkom analizom trenutno dostupnih alata za simuliranje dela za analizu programskog prevodioca. Prikaz karakteristika razvijenog softverskog sistema za vizuelnu reprezentaciju leksičke analize, dat je u trećem poglavlju. U četvrtom poglavlju je opisan način primene alata. Zaključak i planovi za dalje nadograđivanje i razvoj softverskog sistema predstavljeni su u petom poglavlju.

II. OBRAZOVNI INTERAKTIVNI SOFTVERSKI SISTEMI

Obrazovni interaktivni softver omogućava studentima da interaktivno eksperimentišu, pružajući im trenutnu i pouzdanu povratnu informaciju o uspešnosti eksperimenta. Na ovaj način studenti dobijaju priliku da isprobaju različite opcije i trenutno procene stečeno teorijsko znanje. Rezultati brojnih istraživanja ukazuju na pozitivne efekte upotrebe simulacionih sistema.

Pouzdane dokaze da kompjuterske simulacije mogu unaprediti tradicionalne načine edukacije pružaju Rutten, Van Joolingen i Van der Veen u njihovoj studiji [7]. 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. Taher i Khan u radu [8] naglašavaju 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.

Simulacioni alati danas su dostupni skoro za svaku oblast nauke. Efekat učenja teorije verovatnoće uz pomoć simulacionog softvera proučavan je u [9]. Istraživanje je vršeno na grupu od 55 studenata, pri čemu je u njihovoj nastavi korišćen dinamički statistički softver TinkerPlots [10]. Dobijeni nalazi su pokazali da je učenje verovatnoće zasnovano na simulaciji unapredilo veštinu predviđanja i donošenja zaključka kod studenata. Edukacija iz oblasti zaštite životne sredine putem interaktivnih obrazovnih softvera je od izuzetnog značaja [11]. Iskustvo o upotrebi simulacionog softvera za oblast obnovljivih izvora energije predstavljeno je i razmatrano u [12]. 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 postaju opipljivije. Edukativni matematički alat koji svakako zavređuje pažnju je Wolfram Mathematica [13]. 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. Takođe je značajna upotreba obrazovnog simulacionog softvera i u poljoprivredi. Na primer, za proučavanje ponašanja stakleničkih sredina sa različitim konfiguracijama, razvijen je interaktivni, dinamički simulator takvog okruženja [14]. U oblasti genetike Soderberg i Price predlažu edukativni softver Evolve [15]. Web bazirani simulatori za učenje računarskih mreža predstavljeni su u [16], [17]. Edukativni kompjuterski sistem sa web-baziranim simulatorom, dizajniran da pomogne u nastavi i učenju računarske arhitekture, predstavili su Đorđević, Nikolić i Milenković u [18]. 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.

U polju računarskih nauka, upotreba softvera za vizuelizaciju apstraktnih koncepata, naročito dolazi do izražaja kod proučavanja tema iz oblasti programskih prevodioca. Obrada ovih tema je i za profesore izazovan zadatak, s obzirom da se studenti na ovom predmetu, po prvi put susreću sa određenim apstraktnim pojmovima. Studentima je i te kako u interesu da sa razumevanjem savladaju ove teme, jer polje koje se bavi programskim prevodiocima prepliće se s drugim disciplinama, uključujući arhitekturu i organizaciju računara, programske jezike, teoriju formalnih jezika i automata, softverski inženjering i računarsku sigurnost. Pretragom literature može se videti da postoji veći broj autora koji su radili na razvoju simulacionih softvera za edukaciju programskih prevodilaca. U pitanju su različiti softverski sistemi, od jednostavnih sa prilično oskudnim funkcijama do višenamenskih koji nude moćne alate za interaktivnu edukaciju. Na osnovu istraživanja [19], [20] može se videti da se uglavnom radi o simulacionim alatima za učenje teorije formalnih jezika i automata, a zastupljeni su i simulatori za učenje sintaksne analize. Simulatori koji obrađuju ostale teme predmeta programski prevodioci zastupljeni su u manjoj meri, dok se za neke teme ne može naći ni jedan simulator. U vreme pisanja ovog rada nije pronađen ni jedan simulacioni sistem koji obrađuje sve teme predmeta programski prevodioci. U nastavku će biti prikazani samo obrazovni simulacioni softveri koji proučavaju prednji deo programskih prevodilaca, konkretno leksičku, sintaksnu i semantičku analizu. U tabeli 1. navedeni su odabrani softverski sistemi sa osnovnim karakteristikama.

COMVIS Finite Automata je obrazovni softver za vizuelizaciju i simulaciju konačnih automata [21]. Koristeći ovaj softver, konačni automati se mogu definisati u grafičkom uređivaču u obliku dijagrama stanja ili definisanjem funkcije prelaska pomoću tabele tranzicije. Nakon definisanja automata, moguće je pokrenuti vizuelnu simulaciju rada automata za proizvoljan ulazni niz, pri čemu se dobija i tekstualni opis simulacije automata. Implementirana je i opcija simulacije Tompsonovog algoritma.

LLparse i LRparse su dva interaktivna edukativna alata za vizuelizaciju procesa LL i LR parsiranja [22]. Ovi alati se mogu koristiti za objašnjenje postupka generisanja LL(1) i LR(1) sintaksnih tabela kroz niz koraka. 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.

BURGRAM je simulacioni sistem namenjen da olakša nastavu programskih prevodilaca u fazi sintaksne analize [23]. Simulira top-down i bottom-up algoritme analize: LL, SLR, LALR i LR. Tokom simulacije za definisanu gramatiku i zadati ulazni niz, student može pratiti korak po korak postupaka parsiranja, uz kreiranje sintaksne tabele i sintaksnog stabla.

TABELA I. OBRAZOVNI SIMULACIONI SISTEMI IZ OBLASTI PROGRAMSKOG PREVOĐENJA

Simulator	**Univerzitet**	**Programski jezik**	**Platforma**
COMVIS Finite automata	Univerzitet u Prištini, Srbija	Java	Desktop
LLparse & LRparse	Rensselaer Polytechnic Institute, SAD	C++	Desktop
BURGRAM	University of Burgos, Spain	Java	Desktop
PAVT	Netaji Subhas University of Technology, India	C++	Desktop
Webworks applets	Montana State University, SAD	Java	Web
RegExpert	University of Zagreb, Croatia	Nepoznato	Desktop
SELFA-Pro	University of Castilla-La Mancha, Spain	PHP, HTML, XML	Web
LISA	University of Maribor, Slovenia	Java	Desktop
Automata Simulator	Netaji Subhas University of Technology, India	Java	Mobile

PAVT (Parsing Algorithm Visualizer Tool) je simulacioni alat za vizuelizaciju postupka konstrukcije parsera, za zadatu beskonteksnu gramatiku [24]. 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.

Webworks applets predstavljaju rešenje laboratorije Webworks Univerziteta Montana za aktivno učenje tema iz oblasti formalnih jezika i automata [25]. U pitanju su tri apleta, i to aplet za učenje konačnih automata, za učenje regularnih izraza i aplet za učenje regularne gramatike. Simulator ima mogućnost pretvaranja nedeterminističkih konačnih automata u ekvivalentne determinističke, konverziju regularnih izraza ili regularnih gramatika u determinističke konačne automate i obrnuto.

RegExpert interaktivni edukativni softver predstavlja alat za manipulaciju regularnim jezicima [26]. 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 da uoče kako promene utiču na rezultujući automat.

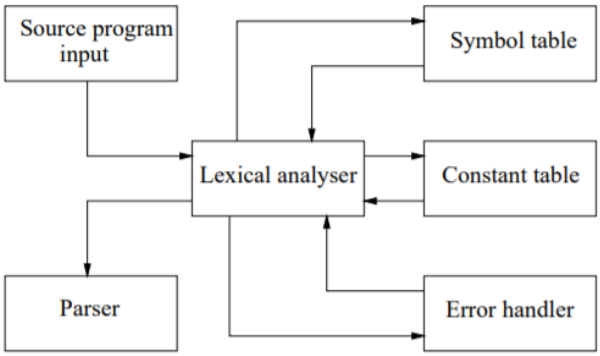
SELFA-Pro (SoftwarE for Learning Formal languages and Automata theory - Pro) je simulacioni edukativni alat osmišljen sa ciljem da unapredi kvalitet nastave [27]. Studenti mogu da eksperimentišu sa gramatikom (regularnom ili beskonteksnom), kao i sa automata (konačni ili push-down). U pitanju je web aplikacija koja razlikuje dve vrste korisnika - studenti i profesori. Profesorima je omogućen pregled aktivnosti studenata.

LISA (Language Implementation System Based on Attribute Grammars) je edukativni sistem koji pruža mogućnost eksperimentisanja i testiranja različitih leksičkih i sintaksnih analizatora [28]. 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.

Automata Simulator je mobilna aplikacija namenjena za simulaciju više tipova automata [29]. Ima vrlo skromne mogućnosti u odnosu na ostale analizirane alate, ali ipak zavređuje pažnju zbog toga što se radi o mobilnoj aplikaciji. Studenti pomoću ove aplikacije mogu da dizajniraju i simuliraju konačne automate, push-down automate, Tjuringovu mašinu, kao i Murovu i Milijevu mašinu.

III. OPIS RADA SISTEMA ZA VIZUELNU REPREZENTACIJU LEKSIČKE ANALIZE

Prva faza u procesu programskog prevođenja je leksička analiza. Leksička analiza programskog jezika podrazumeva analizu izvornog koda, karakter po karakter, identifikujući celine koje se nazivaju lekseme. Prikaz procesa leksičke analize dat je na Sl. 2.



Slika 2. Proces leksičke analize

Za svaku leksemu, leksički analizator kao izlaz proizvodi token oblika:

⟨token-name, attribute-value⟩

koji se prosleđuje sledećoj fazi, to jest sintaksnoj analizi. Prva komponenta tokena token-name je apstraktni simbol koji se koristi tokom sintaksne analize, a druga komponenta attribute-value ukazuje na unos u tabelu simbola za ovaj token. Tabela simbola (engl. symbol table) je struktura podataka u kojoj se, tokom etape analize, prikupljaju informacije o tipu, opsegu i memorijskoj lokaciji identifikatora. Informacije iz tabele simbola potrebne su za semantičku analizu i generisanje koda.

Softverski sistem za vizuelizaciju leksičke analize, pored opisane osnovne funkcije ima i zadatak da studentima na jednostavan način vizuelno prezentuje korake analize u okviru intuitivnog grafičkog okruženja. U pitanju je web bazirana aplikacija, za čiji razvoj su korišćeni jezici PHP, HTML i JavaScript. Razlog za ovakav izbor leži najpre u tome što web bazirane aplikacije imaju veću dostupnost i ne zahtevaju lokalnu instalaciju. Navedeni jezici su izabrani zato što su besplatni za korišćenje i zato što postoji veliki broj razvojnih okruženja koja podržavaju ove jezike. U ovom slučaju korišćen je Eclipse kao razvojno okruženje. Interfejs sistema je dizajniran na engleskom jeziku, ali je ostavljena mogućnost za njegovu internacionalizaciju dodavanjem još podržanih jezika.

Glavni mehanizam za tokenizaciju smešten je u JavaSript fajlu lex.js, koji je tako napisan da se može koristiti i putem konzole, a i po potrebi pozivati iz neke HTML ili PHP stranice. Bitne funkcije ovog JavaSript leksičkog analizatora su:

- compile(rules) i

- next().

Prva funkcija je veoma važna, jer upravo ona čini ovaj alat interaktivnim i dozvoljava korisniku zadavanje sopstvenih regularnih definicija. Web aplikacija obezbeđuje formu za definisanje regularne gramatike. Regularne definicije se nakon toga prosleđuju leksičkom analizatoru na sledeći način:

```
compile({
        WS:      /[ \t]+/,
        comment: /\/\/.*?$/,
        number:  /0|[1-9][0-9]*/,
        lparen:  '(',
        rparen:  ')',
        keyword: ['while', 'if', 'else'],
```

```
        identifier:  /[a-z0-9]+/,
    })
```

Zadatak leksičkog analizatora je da iz ulaznog koda, izbacuje delove koji nisu značajni za sintaksnu analizu (komentare, praznine, tab i newline simbole). Međutim, zbog bolje preglednosti, i razumevanja studenata, iz gornjeg koda se može videti da smo definisali token WS za takve slučajeve.

Druga funkcija se poziva za početak skeniranja ulazne rečenice. Kada funkcija otkrije određeno podudaranje vraća odgovarajući token, i potrebno je ponovo pozivati ovu funkciju sve dok se ne dođe do kraja internog bafera i vrati vrednost "undefined". To znači da je leksički analizator skenirao kompletnu ulaznu rečenicu. Zbog toga se ova funkcija poziva u while petlji, iz koje se izlazi kada bafer dođe do kraja. Za skeniranje nove rečenice, potrebno je pozvati funkciju reset(), kako bi se ispraznio interni bafer. Objekti tokena, koji se vraćaju pozivom funkcije next() su definisani na sledeći način:

```
  var token = {
        type:  (typeof  group.type  ===
'function'  &&  group.type(text))  ||
group.defaultType,
        value:  typeof  group.value  ===
'function' ? group.value(text) : text,
        offset: offset,
        lineBreaks: lineBreaks,
        line: this.line,
        col: this.col,
      }
```

Atributi objekta token imaju sledeće značenje:

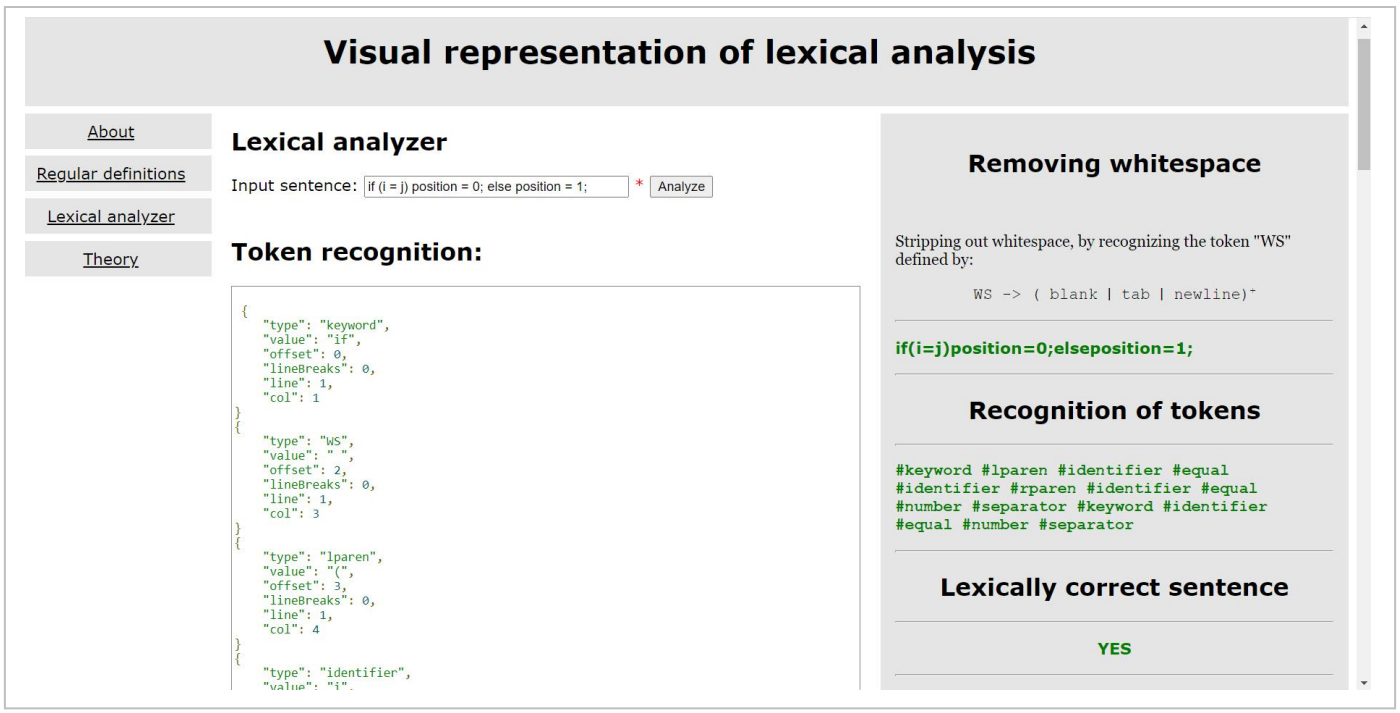
- *type*: Ime definisane grupe tokena, koje je prosleđeno kompajliranju;
- *value*: Vrednost tokena, odnosno string koji odgovara datom tokenu;
- *offset*: Broj bajtova od početka bafera;
- *lineBreaks*: Broj preloma linije;
- *line*: Broj linija od početka ulazne rečenice;
- *col*: Broj pozicije u redu, odnosno kolone u kojoj počinje string za dati token.

Takođe, važna karakteristika ovog leksičkog analizatora je i mogućnost za upravljanje greškama. Izveštavanje o greškama poziva se u slučaju kada leksički analizator naiđe na nepoznate ulazne simbole, ili kada se potkrade greška u osnovnim gramatičkim simbolima. I u tom slučaju, kada tokenizator ne pronađe ni jedno podudaranje sa definicijama, vratiće informaciju o nastaloj grešci, kao i informaciju u tome u kojem redu i na kojoj poziciji, to jest koloni, je nastala greška.

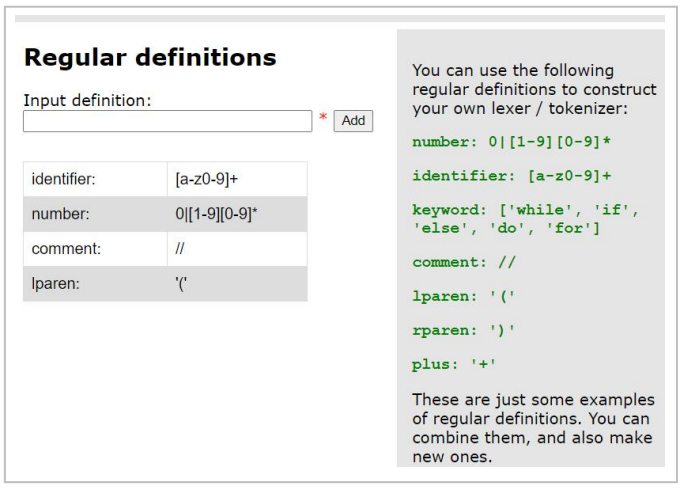
IV. PRIMENA SOFTVERSKOG SISTEMA

Grafički korisnički interfejs web bazirane aplikacije se sastoji od četiri stranice. Prva stranica *about.html* sadrži osnovne informacije o ovom sistemu sa kratkim korisničkim uputstvom. Druga stranica *regular_definitions.php* sadrži formu za definisanje regularnih gramatika. Stranica na kojoj se detaljno vizualizuje postupak leksičke analize je *lexical_analyzer.php* i prikazana je na Sl. 3. Na kraju imamo stranicu *theory.html* koja korisniku aplikacije pruža teorijsku pozadinu leksičke analize.

Dakle, korisnik aplikacije ima mogućnost kreiranja sopstvenog leksičkog analizatora na osnovu unosa željenih regularnih definicija na stranici *regular_definitions.php*, što se može videti na Sl. 4.



Slika 3. Korisnički interfejs stranice *lexical_analyzer.php* na kojoj se može pratiti postupak leksičke analize



Slika 4. Korisnički interfejs stranice regular_definitions.php koja je namenjena za definisanje regularne gramatike

Na prikazanoj slici se može videti da se na desnoj strani korisničkog interfejsa nalaze smernice za definisanje gramatike. Početnici jednostavno mogu iskopirati ove definicije. U datoj formi se unose jedna po jedna definicija klikom na taster *Add*. Dodavanje svake definicije biće vidljivo u tabeli koja se formira odmah ispod forme za unos. Prvi deo definicije predstavlja naziv grupe tokena. Pa tako možemo imati: keywords, identifier, number, comment, operator itd. Navedeni primeri naziva se najčešće koriste, ali aplikacija dozvoljava korisniku da ih po želji sam formira. Takođe, određene grupe tokena se, zavisno od potrebe, mogu različito definisati. Na primer, identifikatori mogu biti definisani na neki od sledećih načina:

- `identifier: [a-zA-Z]+`
- `identifier: [a-z0-9]+`
- `identifier: [a-zA-Z0-9]+`

U okviru korisničkog interfejsa, stranice za vizuelizaciju rada leksičkog analizatora, nalazi se tekstualno polje za unos rečenice, to jest stringa koji će biti prosleđen leksičkom analizatoru. Nakon završenog unosa, izborom opcije *Analyze*, leksički analizator započinje analizu, prezentujući korisniku svaki korak analize. Odmah ispod forme za unos stringa ispisuju se sve neophodne informacije o prepoznatim tokenima, uključujući i WS tokene za praznine, tab i newline simbole. Metodom JSON.stringify(), objekat token vraćen pozivom funkcije next(), se zbog preglednosti konvertuje u JSON string. Na primer, informacije o tokenu za ključnu reč *if*, biće prikazane na sledeći način:

```
{
    "type": "keyword",
    "value": "if",
    "offset": 0,
    "lineBreaks": 0,
    "line": 1,
    "col": 1
}
```

Nakon prikaza detaljnih informacija o prepoznatim tokenima, na desnoj strani korisničkog interfejsa ove stranice prikazuje se sledeći korak leksičke analize. U ovom koraku se, iz ulaznog stringa, vrši uklanjanje delova koji nisu bitni za sintaksnu analizu, komentara, praznina, tab i newline simbola. Tako da će u tom delu biti prikazan ulazni string bez navedenih suvišnih simbola.

Odmah ispod ove sekcije, prikazuje se sledeći korak analize, a to je tokenizacija ulaznog stringa. Na primer, za ulazni string *id1 = id2 - 5;*, biće prepoznat sledeći niz tokena:

```
#identifier#equal#identifier#Error
```

U ovom nizu se može primetiti token *#error* koji ukazuje na pojavu greške, zbog toga što leksički analizator za određeni string ili simbol nije pronašao odgovarajuće podudaranje. Razlog za pojavu greške je operator „-“, jer su aritmetički operatori namerno izostavljeni iz regularnih definicija, zbog demonstriranja greške. I u zadnjoj sekciji na desnoj strani korisničkog interfejsa se ispisuje informacija o tome da li je uneta rečenica leksički ispravna. U slučaju da jeste ispisuje se odgovarajuća informacija o tome i to znači da se identifikovani tokeni sa svim potrebnim informacijama mogu proslediti parseru na sintaksnu analizu. U suprotnom, usled pojave greške, dobijamo informaciju da je ulazna rečenica leksički neispravna i ispisuje se informacija o tačnoj poziciji u rečenici gde je nastala greška.

V. ZAKLJUČAK

Veliki broj teorijskih apstraktnih koncepata iz različitih faza programskog prevođenja može se ilustrovati pomoću simulacionih sistema. U literaturi se može pronaći veći broj takvih simulatora, koji su razvijani na raznim univerzitetima za potrebe primene u nastavnom procesu na njihovim fakultetima. Rezultati brojnih istraživanja ukazuju na pozitivne efekte koje donosi upotreba simulacionih sistema u procesu učenja. U ovom radu su predstavljeni odabrani simulacioni sistemi koji su pogodni za učenje programskih prevodilaca. Neki od ovih sistema ne obrađuju leksičku analizu, dok drugi prikazuju samo nazive tokena generisanih na osnovu ulaznog stringa. Međutim, pored naziva tokena koji se prosleđuju sintaksnom analizatoru, leksički analizator treba da sačuva i neke atribute tih tokena da bi oni bili kasnije iskorišćeni u toku semantičke analize i generisanja koda. Cilj ovog rada bio je usmeren ka realizaciji novog simulacionog sistema za učenje leksičke analize programskih prevodilaca, koji bi mogao da se koristi u procesu obrazovanja. Razvijena je interaktivna web aplikacija koja korisnicima pruža mogućnost definisanja regularnih gramatika, kao i detaljno praćenje svih koraka leksičke analize.

Planovi za buduća istraživanja odnose se na realizaciju softverskog sistema koji bi podržao što veći broj algoritama koji se primenjuju u različitim fazama programskog prevođenja. Poželjno bi bilo da se simulacija i vizuelizacija algoritama vrši u okviru istog web korisničkog interfejsa. Sistem bi trebao da bude modularan, to jest da omogućava jednostavnu nadogradnju dodavanjem novih algoritama. Tako da je sledeći plan, proširenje predstavljenog web baziranog softverskog sistema za modul sintaksne analize.

LITERATURA

[1] G. Beauchamp and S. Kennewell, "Interactivity in the classroom and its impact on learning", Computers and Education, 2010, 54, pp. 759–766.

[2] Q. Wang, "A generic model for guiding the integration of ICT into teaching and learning", Innovations in education and teaching international, 2008, 45(4), pp.411-419.

[3] M. Bétrancourt, "The animation and interactivity principles in multimedia learning", The Cambridge handbook of multimedia learning, 2005, pp.287-296.

[4] E. A. Makarova, "Visual culture in educational environment and innovative teaching technologies", Universal Journal of Management, 2016, 4(11), pp.621-627.

[5] I. Branovic, R. Popovic, N. Jovanovic, R. Giorgi, B. Nikolic and M. Zivkovic, "Integration of simulators in virtual 3D computer science classroom". In 2014 IEEE Global Engineering Education Conference (EDUCON), 2014, pp. 1-4.

[6] P. Tchounikine, Computer science and educational software design: A resource for multidisciplinary work in technology enhanced learning. London: Springer Science & Business Media, 2011.

[7] N. Rutten, W. R. Van Joolingen and J. T. Van der Veen, "The learning effects of computer simulations in science education", Computers & Education, Elsevier, 2012, 58(2012), pp. 136-153.

[8] M. Taher and A. Khan, "Impact of Simulation-based and Hands-on Teaching Methodologies on Students' Learning in an Engineering Technology Program", Proceedings of the ASEE Annual Conference & Exposition, 2014, pp. 1-22.

[9] T. Koparan and G. K. Yilmaz, "The Effect of Simulation-Based Learning on Prospective Teachers' Inference Skills in Teaching Probability", Universal Journal of Educational Research, 2015, 3(11), pp. 775-786.

[10] C. Konold and C. D. Miller, TinkerPlots: Dynamic data exploration. Computer software. Emeryville, CA: Key Curriculum Press, 2005.

[11] A. Williams, K. Lansey and J. Washburne, "A dynamic simulation based water resources education tool", Journal of Environmental Management, 2009, 90(1), pp.471-482.

[12] A. Witzig, M. Prandini, A. Wolf and L. Kunath, "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 2016: International Conference on Solar Energy and Buildings, Spain, 2016, pp. 1591-1600.

[13] S. Wolfram, The Mathematica Book, 5th ed. Illinois: Wolfram Media, Inc, 2003.

[14] E. Fitz-Rodríguez, C. Kubota, G. A. Giacomelli, M. E. Tignor, S. B. Wilson and M. McMahon, "Dynamic modeling and simulation of greenhouse environments under several scenarios: A web-based application". Computers and electronics in agriculture, 2010, 70(1), pp.105-116.

[15] P. Soderberg and F. Price, "An examination of problem-based teaching and learning in population genetics and evolution using EVOLVE, a computer simulation", International Journal of Science Education, 2003, 25(1), pp. 35-55.

[16] N. Jovanović, R. Popović, S. Marković and Z. Jovanovic, "Web laboratory for computer network", Computer Applications in Engineering Education, 2012, 20(3), pp. 493-502.

[17] N. Jovanović, R. Popović and Z. Jovanović, "WNetSim: a web-based computer network simulator". International Journal of Electrical Engineering Education, 2009, 46(4), pp. 383-396.

[18] J. Djordjevic, B. Nikolic and A. Milenkovic, "FlexibleWeb-Based Educational System for Teaching Computer Architecture and Organization", IEEE Transactions on Education, 2005, 48(2), pp. 264-273.

[19] S. Stamenković and N. Jovanović, "Comparative analysis of simulation system for teaching compilers", Bizinfo (Blace), 2019, 10(2), pp. 1–23.

[20] S. Stamenković, N. Jovanović and P. Chakraborty, "Evaluation of simulation systems suitable for teaching courses on compiler construction", Comput. Appl. Eng. Educ. 2020, 28, pp. 606–625.

[21] N. Jovanović, D. Miljković, S. Stamenković, Z. Jovanović and P. Chakraborty, "Teaching concepts related to finite automata using ComVis". Computer Applications in Engineering Education, https://doi.org/10.1002/cae.22353

[22] S. A. Blythe, M. C. James and S. H. Rodger, "LLparse and LRparse: Visual and interactive tools for parsing", ACM SIGCSE Bull, 1994, 26(1), pp. 208–212.

[23] C. García-Osorio, C. Gómez-Palacios and N. García-Pedrajas, "A tool for teaching LL and LR parsing algorithms", Annual Conference on Innovation and Technology in Computer Science Education ITiCSE'08, 2008, p. 317.

[24] S. Sangal et al., PAVT: "A tool to visualize and teach parsing algorithms", Educ. Inf. Technol. 2018, 23(6), pp. 2737–2764.

[25] M. T. Grinder et al., Loving to learn theory: Active learning modules for the theory of computing, ACM SIGCSE Bull. 2002, 34(1), pp. 371–375.

[26] I. Budiselic, S. Srbljic and M. Popovic, "RegExpert: A tool for visualization of regular expressions", EUROCON 2007-The International Conference on Computer as a Tool, 2007, pp. 2387–2389.

[27] J. J. Castro-Schez et al., "Designing and using software tools for educational purposes: FLAT, a case study", IEEE Trans. Educ. 2009, 52(1), pp. 66–74.

[28] M. Mernik, and V. Zumer, "An educational tool for teaching compiler construction", IEEE Trans. Educ. 2003, 46(1), pp. 61–68.

[29] T. Singh et al., Automata simulator: "A mobile app to teach theory of computation", Comput. Appl. Eng. Educ. 2019, 27(5), pp. 1064–1072.

ABSTRACT

In order to improve traditional teaching, visualization tools, simulators, animation systems and other educational software tools for illustration and visualization of abstract concepts are successfully applied in many areas of computer science. In this paper, such systems are analyzed, especially those that are suitable for studying the front part of the program compiler, which includes the phases of analysis. The main goal of this research work is the development of an interactive software system for visualization of lexical analysis in the process of program compilation. It is a web-based application that gives students the ability to define regular grammar, enter a specific string, and track how a lexical analyzer takes a series of characters, groups them into tokens, and produces an output token sequence. This type of interactive experiment with theoretical concepts should simplify complex topics and make them more interesting.

EDUCATIONAL INTERACTIVE TOOL FOR VISUAL REPRESENTATION OF LEXICAL ANALYSIS

Srećko Stamenković, Nenad Jovanović