Softverski alat kao pomoćno sredstvo za učenje leksičke i sintaksne analize

Nenad Jovanović
Fakultet tehničkih nauka
Univerzitet u Prištini
Kosovska Mitrovica, Srbija
nenad.jovanovic@pr.ac.rs

Srećko Stamenković, Miloš Cvjetković, Zoran Jovanović
Odsek Viskoka poslovna škola Blace
Akademija strukovnih studija Južna Srbija
stamenkovic.srecko@gmail.com, miloscvj@gmail.com,
zoranjov2004@gmail.com

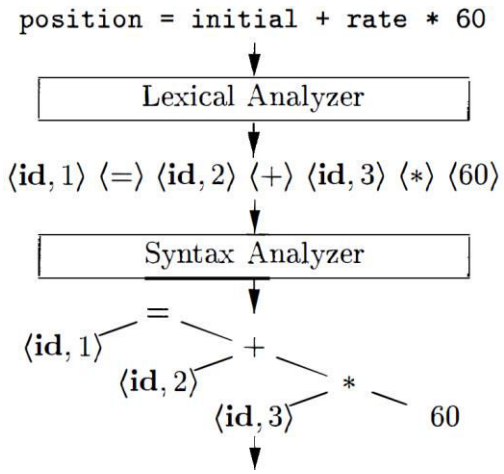
Sažetak— **Zbog visokog nivoa apstrakcije teorije programskog prevođenja, smanjuje se zainteresovanost studenata za učenje tema ovog predmeta. Pristup zasnovan samo na tradicionalnim predavanjima nije motivišući i ne utiče dobro na profesionalni razvoj studenata. Naše pedagoško iskustvo pokazuje da, pored primenjene nastavne metodologije, na ishod učenja kompleksnih teorijskih konstrukcija, u velikoj meri može da utiče i primena adekvatnih softverskih alata kao pomoćnih sredstava u nastavnom procesu. U ovom radu prezentujemo alat koji smo razvili sa ciljem, da studentima približimo proces leksičke, sintaksne i semantičke analize. Softverski alat je predviđen za primenu u odgovarajućim laboratorijskim vežbama. Evaluacija ovog softverskog sistema je izvršena eksperimentalnom metodom za procenu efikasnosti alata i procenom korisničkog iskustva kvantitativnom anketom studenata.**

Ključne reči - programski prevodioci; konstrukcija kompajlera; obrazovni alati; softversko inženjerstvo;

I. UVOD

Konstrukcija kompajlera je jedan od osnovnih predmeta u okviru nastavnog plana studijskog programa računarskih nauka [1]. U pitanju je jednosemestralni kurs koji sadrži važne, ali veoma kompleksne teme. Realizacija bilo kog programskog jezika ne može se odvojiti od tehnologije kompajliranja. Zbog toga je neophodno da student računarstva nauči i savlada osnovnu strukturu i tehnike implementacije kompajlera za dalje učenje, istraživački i stručni rad [2]. Sadržaj ovog kursa, dakle, predstavlja kamen temeljac softverskog inženjerstva, međutim, često je i kamen spoticanja za neke studente. Praćenjem rezultata studenata na ovom predmetu, primećeno je da imaju problema da savladaju gradivo koje obiluje kompleksnim teorijskim konstrukcijama, što se na kraju manifestuje lošim rezultatima na ispitu.

Prevođenje ulaznog programa napisanog na višem programskom jeziku započinje leksičkom analizom. U ovoj fazi ulazni niz izvornog programa se najpre rastavlja na sastavne delove, to jest leksičke jedinice, kako bi se pripremio za sintaksnu analizu. Na osnovu dobijenih leksema generišu se odgovarajući tokeni koji odgovaraju simbolima korišćenog programskog jezika. Dobijeni niz tokena se upućuje u sintaksni analizator gde se vrši provera da li su naredbe u skladu sa formalnim opisom jezika. Uprošćeni prikaz procesa leksičke i sintaksne analize prema [3] može se videti na Sl. 1.



Slika 1. Uprošćeni prikaz procesa leksičke i sintaksne analize

Ukoliko ne postoje greške u sintaksi pokreće se proces parsiranja. Dakle, u delu za analizu programskih prevodilaca vrši se leksička, sintaksna i semantička analiza kao i generisanje međukoda. U delu za sintezu vrši se optimizacija međukoda, generisanje ciljnog koda i eventualna optimizacija koda. Sadržaj predmeta "Programski prevodioci" se može razlikovati na različitim visokoškolskim ustanovama, ali na osnovu ovog pojednostavljenog opisa mehanizma rada kompajlera, mogu se prepoznati osnovne teme koje su uglavnom obuhvaćene nastavnim planom predmeta, to su: teorija formalnih jezika, teorija automata, leksička analiza, sintaksna analiza, semantička analiza, generisanje međukoda, generisanje izlaznog koda i optimizacija.

Uvođenje interaktivnih softverskih alata u obrazovanju može podstaći motivaciju studenata [4]. Pored interaktivnosti, softver koji se koristi u obrazovanju, trebao bi da poseduje još jednu važnu osobinu, koja može doprineti efektivnijem prenosu znanja, a to je mogućnost vizuelizacije predmeta proučavanja. Prema Nahvi [5] edukativni alati koji se koriste u nastavi moraju da zadovolje određene zahteve. Trebali bi da budu intuitivni i jednostavni za korišćenje. 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.

TABELA I. ZASTUPLJENOST TEMA KOJE SE ODNOSE NA LEKSIČKU, SINTAKSNU I SEMANTIČKU ANALIZU

Univerzitet	*Naziv kursa*	*Teme kursa*		*%*
		Leksička, sintaksna i semantička analiza	*Ukupno*	
FTN, Univerzitet u Prištini	Programski prevodioci	3	8	37,5
Stanford University	Compilers	7	18	38,9
University of Oxford	Compilers	3	10	30
Carnegie Mellon University	Compiler Design	3	13	23,1
University of Cambridge	Compiler Construction	5	16	31,3
The University of Newcastle	Compiler Design	3	8	37,5

Kompajleri predstavljaju oblast računarskih nauka, u kojoj se teorijski koncepti, kao što su, teorija automata, teorija grafova, formalni jezici i gramatike uspešno primenjuju u praktičnim implementacijama leksičkih analizatora, parsera, semantičkih analizatora, optimizatora koda itd. Većina postojećih kurseva kompajlera na svetskim univerzitetima deli nastavni plan i program na module (teme) koji odgovaraju fazama kompajliranja. Na osnovu analize koja je prikazana u tabeli 1, utvrđeno je da se trećina materijala kursa (uglavnom preko 30%) odnosi na leksičku, sintaksnu i semantičku analizu. Zbog toga smo razvili softverski alat koji je pogodan za primenu u laboratorijskim vežbama, za praćenje ovih faza kompajliranja. Alat omogućava kreiranje regularne gramatike, praćenje tokenizacije ulaznog stringa, upravljanje leksičkim greškama, proveru sintakse na osnovu definisane gramatike, kao i kreiranje tabele simbola.

Rad je sistematizovan u pet poglavlja. Drugo poglavlje prezentuje odabrane obrazovne softverske alate koji su pogodni za učenje apstraktnih teorijskih koncepata kompajlera. Prikaz karakteristika razvijenog alata koji se može koristiti kao pomoćno sredstvo za učenje leksičke i sintaksne analize, dat je u trećem poglavlju. U četvrtom poglavlju je izvršena evaluacija alata eksperimentalnom metodom za procenu efikasnosti alata i procenom korisničkog iskustva. Zaključak i planovi za dalje nadograđivanje i razvoj predstavljeni su u petom poglavlju.

II. POVEZANI RADOVI

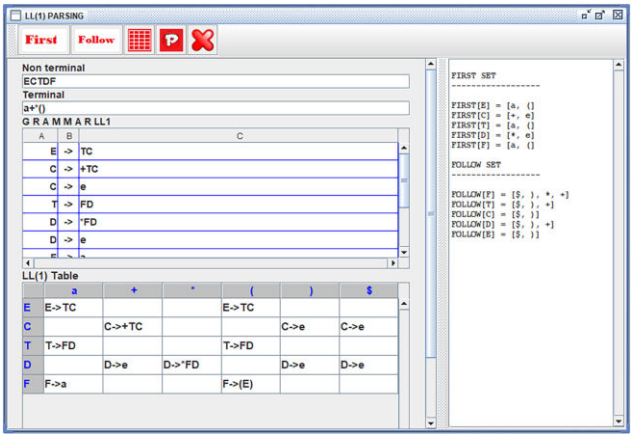
Gotovo svaka faza programskog prevođenja se može ilustrovati pomoću nekog softverskog alata ili simulatora. U literaturi se može pronaći veći broj takvih simulatora, koji su razvijani na raznim univerzitetima za potrebe primene u nastavnom procesu. Uglavnom su razvijani kao desktop ili web aplikacije, ali mogu se pronaći i u vidu mobilnih aplikacija. U pitanju su različiti softverski sistemi, koji se razlikuju i po funkcionalnosti i po fazi kompajliranja koju pokrivaju. Analize pokazuju da se uglavnom radi o alatima i simulatorima za pomoć u učenju teorije automata [6], [7].

ComVis – interaktivno simulaciono okruženje [8] smo razvili kako bismo unapredili tradicionalnu nastavu kompajlera, olakšali praćenje gradiva i time motivisali studente za aktivnije učenje. U pitanju je modularni softverski sistem za vizuelizaciju i simulaciju osnovnih koncepata i algoritama kompajlera. ComVis finite automata je modul koji omogućava studentima da vizuelizuju rad konačnih automata, konvertuju regularne izraze u deterministčke konačne automate (DFA) ili nedeterminističke konačne automate (NFA) i simuliraju Tomsonov algoritam [9], [10]. Simulaciono okruženje sadrži i modul za simuliranje procesa konstruisanja sintaksnih tabela LL(1) i LR(1), i to korak po korak, pri čemu korisnik dobija povratnu informaciju o ispravnosti svakog koraka pre nego što pređe na sledeći (Sl. 2). Sistem, takođe, omogućava izračunavanje FIRST i FOLLOW skupova, grafički prezentuje rad DFA i konstruiše odgovarajuće sintaksno stablo. Ovaj alat obezbeđuje i modul za vizuelizaciju procesa generisanja mašinskog koda.

JFLAP (Java Formal Languages and Automata Package) je interaktivni grafički softverski alat, namenjen za edukaciju formalnih jezika, teorije automata i programskih prevodioca [11], [12]. Razvoj JFLAP-a započeo je 1990. godine najpre u vidu jednostavnije verzije FLAP [13] 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.

Seshat je edukativni alat razvijen na Univerzitetu Burgos, u Španiji. Baziran je na web tehnologiji, namenjen za učenje osnovnih koncepata regularnih jezika, regularnih izraza i konačnih automata [14]. 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.

Automata Simulator je edukativna mobilna aplikacija za simulaciju više tipova automata [15]. Ova aplikacija je razvijena na Netaji Subhas Univerzitetu tehnologije, u Indiji,



Slika 2. ComVis modul za simulaciju procesa parsiranja

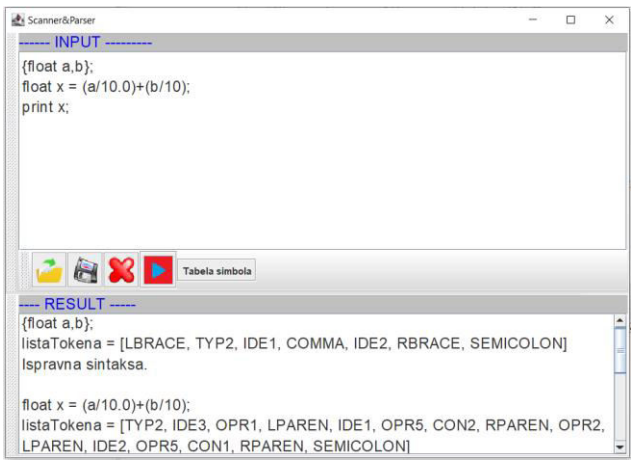
kao pomoćno nastavno sredstvo za podršku studentima u procesu učenja. Studenti pomoću ove aplikacije mogu da dizajniraju i simuliraju konačne automate, push-down automate, Tjuringovu mašinu, kao i Murovu i Milijevu mašinu. Automata Simulator ima vrlo skromne mogućnosti u odnosu na prethodna tri analizirana simulaciona alata, ali ipak zavređuje pažnju zbog toga što se radi o mobilnoj aplikaciji.

III. ALAT ZA PRAĆENJE LEKSIČKE I SINTAKSNE ANALIZE

Alat za praćenje leksičke i sintaksne analize razvili smo sa ciljem da proširimo mogućnosti našeg modularnog simulacionog sistema ComVis. Za razvoj smo koristili Java programski jezik, a kao razvojno okruženje Eclipse. Grafički korisnički interfejs je jednostavan i intuitivan (Sl. 3). Sadrži dva tekstualna polja (Input i Result) i tri tastera (Open, Save, Delete, Run i Tabela simbola).

Leksički analizator u okviru alata čita izvorni kod, napisan u tekstualnom polju Input, karakter po karakter, pri čemu vrši identifikaciju leksičkih jedinica - leksama. Za svaku leksemu se vezuje odgovarajući token. Različiti tokeni su ključne reči, identifikatori, operatori, konstante i znakovi interpunkcije kao što su zarez i zagrade. Na primer lista tokena može biti sledeća:

TYP1 : kljucna rec int
TYP2 : kljucna rec float
PRINT : kljucna rec print
IDE : identifikator, ime promenljive, moze biti int ili float
OPR : operator =,+,-,*,/
LPAREN : simbol '('
RPAREN : simbol ')'
SEMICOLON : simbol ';'
LBRACE : simbol '{'
RBRACE : simbol '}'
COMMA : simbol ','
CON1 : int konstanta
CON2 : float konstanta



Slika 3. Grafički korisnički interfejs softverskog alata

Svaki token je kolekcija karaktera sa dobro definisanim značenjem izvornog programa koju treba tretirati kao jednu celinu. Token se može definisati nizom znakova odnosno šablonom. Šabloni se konstruišu pomoću regularnih izraza. Oni definišu pravila tako da svaka leksema bude prepoznata kao validni token. Na primer, šablon za definisanje identifikatora može biti sledeći:

[A-Za-z_][A-Za-z0-9]*

Leksički analizator analizira uzastopne karaktere u izvornom programu počevši od prvog znaka koji još nije grupisan u token. Da bi se odredio sledeći token, mnogi znakovi se mogu pretraživati izvan sledećeg tokena. Prepoznati tokeni se dalje šalju u parser na analizu sintakse. Sintaksni analizator, ili tako zvani parser, uzima niz tokena iz leksičkog analizatora i proverava da li taj niz pripada jeziku koji je opisan zadatom gramatikom. Primer gramatike:

linijaKoda -> LBRACE deklaracija RBRACE
linijaKoda -> TYP1 IDE OPR1 izraz SEMICOLON
linijaKoda -> TYP2 IDE OPR1 izraz SEMICOLON
linijaKoda -> PRINT IDE SEMICOLON
linijaKoda -> izraz SEMICOLON
deklaracija -> TYP1 identifikator
deklaracija -> TYP2 identifikator
identifikator -> IDE
identifikator -> IDE COMMA identifikator
izraz -> terminal
izraz -> izraz OPR1 izraz
izraz -> izraz OPR2 izraz
izraz -> izraz OPR3 izraz
izraz -> izraz OPR4 izraz
izraz -> izraz OPR5 izraz
izraz -> LPAREN izraz RPAREN
izraz -> LPAREN izraz RPAREN OPR2 izraz
izraz -> LPAREN izraz RPAREN OPR3 izraz
izraz -> LPAREN izraz RPAREN OPR4 izraz
izraz -> LPAREN izraz RPAREN OPR5 izraz
terminal -> IDE
terminal -> CON

Parser uzima zadati kod red po red i proverava sintaksu svakog izraza. Na primer za zadati ulaz:

float x = (a/10.0)+(b/10);

u delu Result alata biće ispisan sledeći rezultat:

listaTokena = [TYP2, IDE3, OPR1, LPAREN, IDE1, OPR5, CON2, RPAREN, OPR2, LPAREN, IDE2, OPR5, CON1, RPAREN, SEMICOLON]

Ispravna sintaksa.

Kao što se može videti, leksički analizator je generisao odgovarajuću listu tokena na osnovu zadatog ulaza, i ta lista je prikazana u okviru uglastih zagrada. Nakon toga, sintaksni analizator je izvršio proveru ispravnosti sintakse zadatog izraza. S obzirom da je u ovom slučaju sintaksa izraza u skladu sa definisanom gramatikom, u delu razultata, alat će ispisati poruku “ispravna sintaksa”.

TABELA II. TABELA SIMBOLA ZA ZADATI ULAZ

a	IDE1	float
b	IDE2	float
10.0	CON2	float
10	CON1	int
x	IDE3	float

Pored praćenja procesa leksičke i sintaksne analize softverski alat omogućava i kreiranje tabele simbola. Tabela simbola 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. Ukoliko na ulazu analizatora zadamo sledeći kod:

```
{float a,b};
float x = (a/10.0)+(b/10);
print x;
```

a zatim odaberemo opciju "Tabela simbola", onda će u delu za prikaz rezultata alat generisati informacije o upotrebljenim identifikatorima, ali i o numeričkim i string konstantama, kao što je to prikazano u tabeli 2.

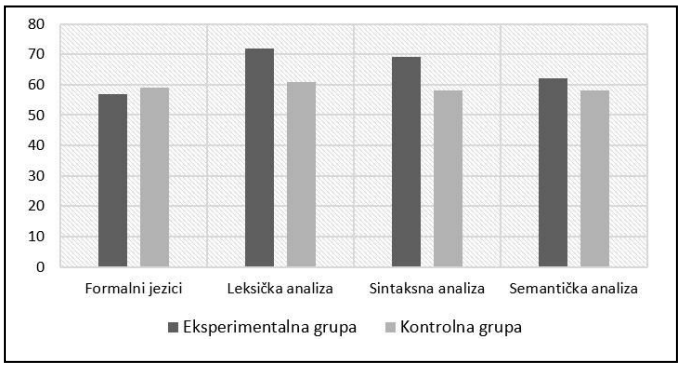
Nakon sintaksne analize sledi semantička analiza. Semantički analizator koristi sintaksno stablo i informacije iz tabele simbola da proveri semantičku usklađenost izvornog programa sa definicijom jezika. U našem primeru, zadati izraz je semantički ispravan i alat će u delu za rezultat ispisati sledeće:

```
Pokusaj dodele float na float.

Semanticki ispravno.
```

IV. EVALUACIJA SOFTVERSKOG ALATA

Za potrebe ovog rada izvršena je evaluacija realizovanog softverskog alata kako bi se procenila njegova efikasnost u oblastima za koje je dizajniran. Evalauacija je izvršena eksperimentalnom metodom za procenu efikasnosti alata i procenom korisničkog iskustva kvantitativnom anketom studenata. Slične metode evaluacije primenjene su u [16] za procenu jednostavnog alata za vizuelizaciju teorijskih koncepata.



Slika 4. Rezultati testiranja dve grupe studenata

TABELA III. REZULTAT ANKETE STUDENATA

U kojoj meri je primena ovog alata unapredila vaše razumevanje iz oblasti:	*Srednja vrednost*
Formalne gramatike	2,33
Leksičke analize	3,78
Sintaksne analize	3,44
Semantičke analize	2,67

Kod eksperimentalne metode koriste se dve grupe studenata, eksperimentalna grupa i kontrolna grupa. U našem slučaju, formirali smo dve grupe od po 9 studenata, pri čemu smo vodili računa o ravnomernoj zastupljenosti prosečnih i naprednih studentata u obe grupe. Obe grupe su pratile predavanja formalnih jezika i gramatika, leksičke, sintaksne i semantičke analize, s tom razlikom što je eksperimentalna grupa imala priliku da koristi tokom predavanja softverski alat kao pomoćno sredstvo. Nakon završenih predavanja obe grupe su dobile isti test koji sadrži pitanja iz oblasti koje su bile izučavane. Rezultati testa prikazani na Sl. 4, pokazuju ohrabrujuće rezultate. Upoređivanjem rezultata može se uočiti neznatni napredak eksperimentalne grupe kod pitanja iz oblasti semantičke analize, dok je situacija kod leksičke i sintaksne analize primetno bolja.

Nakon završenih predavanja studentima eksperimentalne grupe je podeljen i evaluacioni formular. Od studenata je zatraženo da rangiraju uticaj primenjenog alata na njihovo razumevanje tema iz oblasti koje su proučavane na predavanjima. Ocene koje su mogli da daju su: 1 – alat nije doprineo razumevanju teme; 2 – veoma mali stepen doprinosa; 3 – mali stepen doprinosa; 4 – visok stepen doprinosa; 5 – veoma visok stepen doprinosa. Rezultati su prikazani u tabeli 3. Prosečne ocene korisničkog iskustva pokazuju da alat, po mišljenju studenata najviše može da doprinese savladavanju tema iz oblasti leksičke analize.

V. ZAKLJUČAK

Doprinos i značaj ovog rada ogleda se u unapređenju nastavnog procesa, primenom realizovanog edukativnog alata u okviru laboratorijskih vežbi na predmetu „Programski prevodioci". Ovakav način eksperimentisanja sa teorijskim konceptima studentima povećava motivaciju i zainteresovanost za kompleksne teme. Alat koji smo predstavili u ovom radu doprinosi lakšem praćenju i razumevanju procesa leksičke i sintaksne analize. Rezultati sprovedene evaluacije upravo to i potvrđuju. Ovaj alat se može koristiti i za demonstraciju određenih tema iz oblasti regularnih gramatika i semantičke analize.

Iako su rezultati evaluacije pokazali zadovoljavajuće rezultate, softverski alat ipak treba usvršavati u pogledu funkcionalnosti, vizuelnog prikaza, ali i interaktivnosti sa korisnikom. Plan za dalji razvoj, pored navedenog usavršavanja, podrazumevao bi i integraciju ovog alata kao posebnog modula u okviru simulacionog sistema ComVis.

LITERATURA

[1] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, Computer Science

Curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science, available at https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf

[2] N. Wang, and S. Zhang. "Construction of compiler technology course in application-based university." In 2013 the International Conference on Education Technology and Information System (ICETIS 2013), pp. 633-636. Atlantis Press, 2013.

[3] A. V. Aho, M. S. Lam, R. Sethi and J. D. Ullman, Compilers: Principles, techniques, and tools, second edition. Boston: Pearson Education, Inc, 2007.

[4] C. Evans and N. J. Gibbons, "The interactivity effect in multimedia learning," Computers & Education, vol. 49, no. 4, pp.1147-1160, 2007.

[5] M. Nahvi, "Dynamics of student-computer interaction in a simulation environment: Reflections on curricular issues," Proceedings of the IEEE Frontiers in Education FIE'96 26th Annual Conference, vol. 3, USA, pp. 1383-1386, 1996.

[6] S. Stamenković and N. Jovanović, "Comparative analysis of simulation system for teaching compilers," Bizinfo (Blace), vol. 10, no. 2, pp. 1–23, 2019.

[7] S. Stamenković, N. Jovanović and P. Chakraborty, "Evaluation of simulation systems suitable for teaching courses on compiler construction," Comput. Appl. Eng. Educ. vol. 28, no. 3, pp. 606–625, 2020.

[8] N. Jovanović, S. Stamenković, D. Miljković, and P. Chakraborty, "ComVIS—Interactive simulation environment for compiler learning", Comput. Appl. Eng. Educ. (2021), 1–17. https://doi.org/10.1002/cae.22456

[9] N. Jovanović, D. Miljković, S. Stamenković, Z. Jovanović, and P. Chakraborty, "Teaching concepts related to finite automata using ComVis", Comput. Appl. Eng. Educ. (2020). https://doi.org/10.1002/cae.22353

[10] N. Jovanović, S. Stamenković, and D. Miljković, "Vizuelni i interaktivni alat za učenje konačnih automata", 19th International Symposium INFOTEH - JAHORINA 18 - 20 March 2020, VRT.1 (109).

[11] 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, pp. 121-125, 1996.

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

[13] M. LoSacco and S. Rodger, "FLAP: A tool for drawing and simulating automata," EDMEDIA 93, pp. 310-317, 1993.

[14] 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," Comput. Appl. Eng. Educ. vol. 26, no. 6, pp. 2255-2265, 2018.

[15] T. Singh, S. Afreen, P. Chakraborty, R. Raj, S. Yadav and D. Jain, "Automata Simulator: A mobile app to teach theory of computation," Comput. Appl. Eng. Educ. vol. 27, no. 5, pp.1064-1072, 2019.

[16] E. Johnsen, M. Nilsen, E. Hjelseth and C. Merschbrock. Exploring a simple visualization tool for improving conceptual understanding of classical beam theory. Procedia engineering 164:172-179, 2016.

ABSTRACT

Due to the high level of abstraction of compiler theory, students' interest in learning the topic of this subject decreases. An approach based only on traditional lectures is not motivating and does not have a good effect on the professional development of students. Our pedagogical experience shows that, in addition to the applied teaching methodology, the learning outcome of complex theoretical constructions can be greatly influenced by the use of adequate software tools as aids in the teaching process. In this paper, we present a tool that we have developed with the aim of explaining to students the process of lexical, syntax and semantic analysis. This software tool is intended for use in appropriate laboratory exercises. The evaluation of this software system was performed by an experimental method for assessed the efficiency of the tool and evaluating the user experience by a quantitative student survey.

SOFTWARE TOOL AS AN AID TO LEARNING LEXICAL AND SYNTAX ANALYSIS

Nenad Jovanović, Srećko Stamenković, Miloš Cvjetković, Zoran Jovanović