ELEKTROTEHNIŠKI VESTNIK 85(1-2): 49–53, 2018
IZVIRNI ZNANSTVENI ČLANEK

O optimalnosti izračuna semantičnih pravil po strategiji od-zgoraj-navzdol

Boštjan Slivnik

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana, Slovenija
E-pošta: bostjan.slivnik@fri.uni-lj.si

Povzetek. Pri implementaciji sintaksno usmerjenega prevajanja na podlagi kontekstno neodvisnih gramatik pogosto stremimo k (implicitni) gradnji drevesa izpeljav po strategiji od-zgoraj-navzdol, pri čemer poddrevesa gradimo rekurzivno od leve proti desni. V primerjavi z drugimi ta strategija omogoča naravnejši zapis semantičnih pravil in učinkovitejše reševanje iz napak, pri teoretični obravnavi sintaksne analize pa ji ustreza izračun leve sledi izpeljave. V tem članku je definiran kriterij za optimalnost izračuna leve sledi izpeljave glede na količino prepletanja izvajanja posameznih korakov sintaksne analize in izpisa posameznih produkcij leve sledi izpeljave; to ustreza izračunu semantičnih pravil. Prepletanje je v določenih primerih celo nujno potrebno za pravilno delovanje samega sintaksnega analizatorja, hkrati pa pogosto še izboljša reševanje iz napak in tvorjenje smiselnih obvestil o napakah. Kriterij je v članku uporabljen za primerjavo nekaterih najpomembnejših sodobnih algoritmov sinataksne analize, pri čemer pokažemo, da le redki algoritmi podpirajo popolno prepletanje.

Ključne besede: sintaksna analiza, leva sled izpeljave, prepletanje izračuna, kriterij optimalnosti

On the optimal top-down evaluation of semantic rules

In syntax-directed translation based on context-free grammars, a top-down construction of a derivation tree, where subtrees are constructed from left to right, is often preferred to other strategies. It makes formulation of semantic rules simpler and provides a better foundation for error recovery. In the parsing theory, it is modelled by computation producing the left parse of the input string derivation. In this paper, a criterion for the optimal printout of the left parse is defined regarding the interleaving of parser actions and printout of individual productions comprising the resulting left parse. In some cases, parsing cannot be done at all without at least some semantic rule evaluation, while in most cases interleaving of parsing and semantic rule evaluation can significantly improve error recovery and the quality of error messages. Finally, the criterion is applied to some of the most important contemporary parsing algorithms. It is shown that not all algorithms support full interleaving.
Keywords: parsing, left parse, computation interleaving, optimality critetion

1 UVOD

Sintaksni analizatorji na podlagi kontekstno neodvisnih gramatik se večinoma, še zlasti pa pri prevajanju programskih jezikov, uporabljajo kot osnova za implementacijo sintaksno usmerjenega prevajanja [1]. Za opis letega se ponavadi uporabljajo atributne gramatike [1], [2], pri katerih simbolom gramatike pripišemo atribute, v produkcije pa vstavimo semantična pravila za izračun vrednosti atributov. Med sintaksno usmerjenim prevajanjem izračunane vrednosti atributov sestavljajo prevod.

Prejet 25. oktober, 2017
Odobren 13. december, 2017

Semantična pravila lahko opisujejo zgolj graditev abstraktnega sintaksnega drevesa, lahko pa vsebujejo tudi druge, zahtevnejše izračune. Ti omogočajo boljše reševanje iz napak, lahko pa tudi povratno vplivajo na sintaksno analizo, kar je v nekaterih primerih celo nujno za njen pravilni potek.

Trenutek izračuna semantičnega pravila je določen z mestom pravila v produkciji in s samim algoritmom sintaksne analize. Zato morajo biti semantična pravila vstavljena na točno določena mesta: če se izračun semantičnega pravila sproži prekmalu, sintaksni analizator morda sploh še ni prebral vseh za izračun potrebnih vhodnih simbolov, če pa se sproži prepozno, ne more podpreti morebitnega reševanja iz napak in/ali povratno vplivati na sintaksno analizo.

V zadnjem času se je pojavilo nekaj novih algoritmov sintaksne analize [3], [4], [5], [6], [7], [8], [9], pri katerih tako kot pri LL analizi vrstni red izračuna semantičnih pravil ustreza obhodu (implicitno zgrajenega) drevesa izpeljav po strategiji od-zgoraj-navzdol (in od-leve-proti-desni), le da vsi presegajo desetletja staro omejitev na LL gramatike. Ponavadi se avtorji pri primerjavi osredotočijo na razred gramatik, za katere so posamezni algoritmi primerni, v tem delu pa nas bo zanimala primerjava glede na fleksibilnost izračuna semantičnih pravil med samim potekom sintaksne analize.

2 OSNOVNI POJMI IN NOTACIJA

Uporabljeno je uveljavljeno izrazoslovje teorije formalnih jezikov in sintaksne analize, notacija pa je vzeta iz del Sippuja in Soisalon-Soininena [10], [11]. Od bralca

se pričakuje poznavanje LL in LR sintaksne analize.

Množica A^* vsebuje vse nize končne dolžine, ki so sestavljeni iz elementov množice A. Prazen niz označimo z ε. Zapisi $|w|$, w^{R} in w^n označujejo dolžino niza w, obratni niz niza w in n-kratno ponovitev niza, $k{:}w$ pomeni prefiks niza w dolžine k (ali kar w če je $|w| \leq k$). Zapis $u \sqsubset w$ (oz. $u \sqsubseteq w$) pove, da je niz u pravi podniz (oz. podniz ali enak) niza w.

Naj končni množici N in T, za kateri velja $N \cap T = \emptyset$, vsebujeta vmesne in končne simbole, zaporedoma, in naj končna množica $P \subset N \times (N \cup T)^*$ vsebuje produkcije, ki jih pišemo v obliki $[A \longrightarrow \alpha]$ pri $\langle A, \alpha \rangle \in P$. Tedaj je $G = \langle N, T, P, S \rangle$ kontekstno neodvisna gramatika z začetnim simbolom $S \in N$.

Skrajno leva izpeljava po produkciji $[A \longrightarrow \alpha]$ gramatike G je definirana kot relacija

$$(\Longrightarrow^{p}_{G,\mathrm{lm}}) = \{\langle xA\delta, x\alpha\delta \rangle \ ; \ x \in T^* \wedge \delta \in (V \cup T)^*\}.$$

Izpeljava po levi sledi $\pi \in P^*$ je definirana kot

$$(\Longrightarrow^{\varepsilon}_{G,\mathrm{lm}}) = \mathrm{id}_{V^*}$$
$$(\Longrightarrow^{\pi p}_{G,\mathrm{lm}}) = (\Longrightarrow^{\pi}_{G,\mathrm{lm}}) \circ (\Longrightarrow^{p}_{G,\mathrm{lm}})$$

Če sled izpeljave ni pomembna, v znaku za relacijo izpeljave po levi sledi uporabimo * namesto imena sledi. Ekvivalentno definiramo tudi skrajno desno izpeljavo po produkciji in po desni sledi izpeljave. Jezik kontekstno neodvisne gramatike G je definiran kot $L(G) = \{w \in T^* \ ; \ \exists \pi \in P^* \colon S \Longrightarrow^{\pi}_{G,\mathrm{lm}} w\}$.

3 KRITERIJ OPTIMALNOSTI

Da premostimo razliko med teoretično obravnavo sintaksne analize in sintaksno usmerjenim prevajanjem, definiramo na podlagi gramatike $G = \langle N, T, P, S \rangle$ gramatiko $\bar{G} = \langle \bar{N}, T, \bar{P}, S \rangle$ z množico vmesnih simbolov

$$\bar{N} = N \cup \{p_i \ ; \ p = [A \longrightarrow \alpha] \in P \wedge 0 \leq i \leq |\alpha|\}$$

in množico produkcij

$$\bar{P} = \{[A \longrightarrow p_0 X_1 p_1 X_2 p_2 \ldots p_{n-1} X_n p_n] \ ; \ p = [A \longrightarrow X_1 X_2 \ldots X_n] \in P\} \cup \{[p_i \longrightarrow \varepsilon] \ ; \ p_i \in \bar{N} \setminus N\}$$

Produkcije $[p_i \longrightarrow \varepsilon]$ nastopajo v gramatiki $\bar{G}$ na mestih, kjer lahko nastopajo semantična pravila v gramatiki G: v trenutku, ko se med sintaksno usmerjenim prevajanjem na podlagi gramatike G sproži semantično pravilo na mestu p_i, sintaksni analizator za $\bar{G}$ izpiše produkcijo $[p_i \longrightarrow \varepsilon]$ na izhod. (Ta pristop je v jedru zelo podoben kontekstno neodvisnim prevodnim gramatikam [12], le da kljub preprostejšemu zapisu na tem mestu povsem zadošča.)

Primer 1: Dana naj bo gramatika G_1 s produkcijami $p_1 = [S \longrightarrow aABb]$, $p_2 = [A \longrightarrow a]$ in $p_3 = [B \longrightarrow b]$. Pripadajoča gramatika $\bar{G}_1$ vsebuje produkcije

$$\bar{p}_1 = [S \longrightarrow p_{1,0}\, a\, p_{1,1}\, A\, p_{1,2}\, B\, p_{1,3}\, b\, p_{1,4}],$$
$$\bar{p}_2 = [A \longrightarrow p_{2,0}\, a\, p_{2,1}],$$
$$\bar{p}_3 = [B \longrightarrow p_{3,0}\, b\, p_{3,1}] \text{ in}$$
$$\bar{p}_{i,j} = [p_{i,j} \longrightarrow \varepsilon]$$

za vse simbole $p_{i,j}$, ki nastopajo v $\bar{p}_1$, $\bar{p}_2$ in $\bar{p}_3$.

Izhod LL(1) analizatorja za gramatiko $\bar{G}_1$ tik pred pomikom drugega a-ja na sklad vsebuje levo sled

$$\bar{\pi}_1 = \bar{p}_1\, \bar{p}_{1,0}\, \bar{p}_{1,1}\, \bar{p}_2\, \bar{p}_{2,0},$$

v kateri produkcije $\bar{p}_{1,0}$, $\bar{p}_{1,1}$, in $\bar{p}_{2,0}$ povedo, katera semantična pravila in v kakšnem vrstem redu bi se izvedla med sintaksno usmerjenim prevajanjem, ki bi temeljilo na gramatiki G_1 in sintaksni analizi vrste LL(1). □

Za gramatike brez nekoristnih simbolov lahko takoj zapišemo naslednje tri ugotovitve:

Trditev 1: $L(\bar{G}) = L(G)$. (Dokaz očiten.) □

Trditev 2: $\forall k \geq 0 \colon G \in \mathrm{LL}(k) \Longrightarrow \bar{G} \in \mathrm{LL}(k)$.

Dokaz: Noben simbol $p_i \in \bar{N} \setminus N$ ne more povzročiti LL(k) nasprotja, saj zanj obstaja ena sama produkcija.

Predpostavimo, da v $\bar{G}$ obstajata produkciji $\bar{p}_1$ in $\bar{p}_2$, ki obe razvijata isti simbol $A \in \bar{N} \cap N$ in sta napovedani na isto vsebino vhodnega okna x. Tedaj se na isti x prožita tudi izvirni produkciji p_1 in p_2, kar je v nasprotju s predpostavko $G \in \mathrm{LL}(k)$. □

Trditev 3: $\forall k \geq 0 \colon G \in \mathrm{LL}(k) \Longrightarrow \bar{G} \in \mathrm{LR}(k)$.

Dokaz: Po trditvi 2 velja $\bar{G} \in \mathrm{LL}(k)$, torej velja tudi $\bar{G} \in \mathrm{LR}(k)$ ([11], izrek 8.57). □

Kot je omenjeno že v uvodu, se izračun semantičnega pravila ne sme začeti, dokler sintaksni analizator ni prebral vseh vhodnih simbolov, katerih atributi nastopajo v pravilu.

Primer 2: Vzemimo ponovno gramatiko G_1 iz primera 1. Hipotetični sintaksni analizator bi lahko takoj na začetku analize izpisal levo sled $p_1 p_2 p_3$, potem pa svojo napoved preveril tako, kot svoje napovedi preverja na primer LL(1) analizator. Enak sintaksni analizator za $\bar{G}_1$ bi napovedal levo sled

$$\bar{p}_1\, \bar{p}_{1,0}\, \bar{p}_{1,1}\, \bar{p}_2\, \bar{p}_{2,0}\, \bar{p}_{2,1}\, \bar{p}_{1,2}\, \bar{p}_3\, \bar{p}_{3,0}\, \bar{p}_{3,1}\, \bar{p}_{1,3}\, \bar{p}_{1,4}.$$

A takoj na začetku sintaksno usmerjenega prevajanja, torej pred branjem prvega a-ja, se lahko izvede le semantično pravilo na mestu produckije $\bar{p}_{1,0}$ — že pravilo na mestu $\bar{p}_{1,1}$ zahteva prvi a (če ga ne, naj ga snovalec pridruži pravilu $\bar{p}_{1,0}$). Napoved preostanka je s stališča sintaksno usmerjenega prevajanja brez smisla. □

Da zagotovimo opis smiselnega prefiksa leve sledi pri že prebranem prefiksu vhodne besede, definiramo relacijo *skrajno leve izpeljave prefiksa u po produkciji* $p = [A \longrightarrow \alpha]$ kot

$$(\Longrightarrow^{p}_{G,u:\mathrm{lm}}) = \{\langle xA\delta, x\alpha\delta \rangle \in (\Longrightarrow^{p}_{G,\mathrm{lm}}) \ ; \ x \sqsubseteq u\}$$

in jo po običajni poti posplošimo še na izpeljave po levi sledi π. Če je jasno iz konteksta, oznako gramatike izpustimo.

Primer 3: Vrnimo se k primeru 2. Tik pred branjem prvega simbola je prebrani prefiks vhodne besede enak ε. Najdaljša izpeljava za prefiks ε po gramatiki G_1 je $S \Longrightarrow^{p_1}_{\varepsilon:\mathrm{lm}} aABb$, po gramatiki $\bar{G}_1$ pa

$$\begin{aligned} S &\Longrightarrow^{\bar{p}_1}_{\varepsilon:\mathrm{lm}} p_{1,0}\, a\, p_{1,1}\, A\, p_{1,2}\, B\, p_{1,3}\, b\, p_{1,4} \\ &\Longrightarrow^{\bar{p}_{1,0}}_{\varepsilon:\mathrm{lm}} a\, p_{1,1}\, A\, p_{1,2}\, B\, p_{1,3}\, b\, p_{1,4}, \end{aligned}$$

kar ustreza izvedbi semantičnega pravila na mestu produkcije $\bar{p}_{1,0}$.

Podobno velja za veljavni prefiks a. Tik pred pomikom drugega a-ja je prebrani prefiks zgolj a, zato izpeljava po relaciji $(\Longrightarrow^{*}_{a:\mathrm{lm}})$ ustreza ravno levi sledi $\bar{\pi}_1$ iz primera 1:

$$S \Longrightarrow^{\bar{\pi}_1}_{a:\mathrm{lm}} aa\, p_{2,1}\, p_{1,2}\, B\, p_{1,3}\, b\, p_{1,4}.$$

To pomeni, da se bodo izvedla semantična pravila na mestu produkcij $\bar{p}_{1,0}$, $\bar{p}_{1,1}$ in $\bar{p}_{2,0}$, za kaj več pa mora sintaksni analizator prebrati naslednji a z vhoda. □

Predpostavimo, da je sintaksni analizator že prebral prefiks u, niz z dolžine k pa je v vhodnem oknu. Leva sled izpeljave poljubne vhodne besede, ki se začne s prefiksom uz, se lahko začne s katerokoli levo sledjo iz množice

$$\Pi_{u,z} = \left\{\pi_u \;;\; S \Longrightarrow^{\pi_u}_{u:\mathrm{lm}} u\delta \wedge z \in \mathrm{FIRST}_k(\delta)\right\}.$$

To pomeni, da lahko ob prebranem prefiksu u in vsebini vhodnega okna z izhod sintaksnega analizatorja vsebuje kvečjemu najdaljši skupni prefiks

$$\pi_{u,z} = \max\left\{\pi \in P^* \;;\; \forall \pi_u \in \Pi_{u,z} \colon \pi_u \sqsubseteq \pi\right\}$$

vseh levih sledi iz množice $\Pi_{u,z}$. Če bi vseboval daljšo levo sled od $\pi_{u,z}$, bi to lahko pri nekaterih sufiksih v, pri $z \sqsubseteq v$, vodilo do napak.

Zato pravimo, da sintaksni analizator z vhodnim oknom dolžine k *optimalno izpiše levo sled izpeljave po gramatiki* G natanko tedaj, ko njegov izhod za vsak prefiks u poljubne vhodne besede $w = uv \in L(G)$ vsebuje prefiks $\pi_{u,z}$ pri $z = k{:}v$.

Definicija optimalnosti izpisa leve sledi izpeljave seveda ni vezana zgolj na LL gramatike in sintaksno analizo vrste LL.

Primer 4: Vzemimo gramatiko $G_4 \notin \mathrm{LL}$ s produkcijami $[S \longrightarrow Ab]$, $[A \longrightarrow Aa]$ in $[A \longrightarrow a]$. Za vsak $i \geq 0$ obstaja izpeljava

$$S \Longrightarrow^{[S\longrightarrow Ab]}_{a:\mathrm{lm}} Ab \Longrightarrow^{[A\longrightarrow Aa]^i}_{a:\mathrm{lm}} Aa^i b \Longrightarrow^{[A\longrightarrow a]}_{a:\mathrm{lm}} a^{i+1} b,$$

kar pomeni, da velja

$$\Pi_{a^n,a} = \{[S \longrightarrow Ab][A \longrightarrow Aa]^i[A \longrightarrow a] \;;\; i \geq n\}$$

in

$$\pi_{a^n,a} = [S \longrightarrow Ab][A \longrightarrow Aa]^n.$$

Skratka, ko je analizator prebral a^n in je v vhodnem oknu a, se leva sled izpeljave vhodnega niza gotovo začne s $\pi_{a^n,a}$. A ker je na vhodu lahko beseda $a^m b$ pri $m > n+1$, analizator še ne sme izpisati produkcije $[A \longrightarrow a]$. □

Definicija optimalnega izpisa leve sledi seveda velja tudi za pripadajočo gramatiko $\bar{G}$, kar pomeni, da lahko na podlagi te definicije ocenimo ustreznost posameznega sintaksnega analizatorja za izračun semantičnih pravil.

4 OCENA RAZLIČNIH METOD IZRAČUNA LEVE SLEDI

4.1 Sintaksna analiza vrste LL

LL analiza je standardni postopek za izračun leve sledi izpeljave [13], [11]. Osnovni rezultat LL analize pravi, da pri analizi vhodnega niza $w = uv$, za katerega po gramatiki G obstaja izpeljava

$$S \Longrightarrow^{\pi_u}_{\mathrm{lm}} uA\delta \Longrightarrow^{[A\longrightarrow\alpha]}_{\mathrm{lm}} u\alpha\delta \Longrightarrow^{\pi_v}_{\mathrm{lm}} uv = w,$$

vsebuje izhod sintaksnega analizatorja vrste LL(k) v trenutku, ko je prebran prefiks u in je niz $k{:}v$ v vhodnem oknu, levo sled π_u ([11], lemi 8.32 in 8.33). Povedano drugače, vsakič, ko je simbol A na vrhu sklada LL analizatorja, ta napove in izpiše produkcijo $[A \longrightarrow \alpha]$; izhod zato vsebuje levo sled $\pi_u[A \longrightarrow \alpha]$.

Ker $w \in T^*$, obstaja taka izpeljava gornje oblike, da velja bodisi $\alpha = a\alpha'$ pri $a \in T$ bodisi $\alpha\delta = \varepsilon$. Tedaj velja $\pi_{u,z} = \pi_u[A \longrightarrow \alpha]$ pri $z = k{:}v$.

LL analizator torej vedno optimalno izpiše levo sled. To je seveda pričakovano, saj je dobro znano, da se pri LL analizi vsako semantično pravilo izvede v trenutku, ko sintaksna analiza doseže njegovo mesto v gramatiki.

4.2 Schmeiser-Barnardov analizator vrste LR

Kot odgovor na znane težave LL analize z levo rekurzijo sta Schmeiser in Barnard leta 1995 predstavila sintaksni analizator, ki temelji na LALR analizi, a kljub temu izračuna levo sled izpeljave [3], njun pristop pa deluje tudi za LR analizo.

Schemiser-Barnardov LR analizator na vsakem koraku skupaj z LR stanjem, ki ustreza veljavnemu prefiksu γX, na sklad potisne še levo sled izpeljave podniza, ki se v vhodni besedi razvije iz simbola X:

- ko LR analizator v stanju $[\gamma]$ opravi pomik simbola $a \in T$, na sklad potisne $[\gamma a]_\varepsilon$;
- ko LR analizator v stanju $[\gamma]$ opravi prevedbo po produkciji $[A \longrightarrow X_1 X_2 \ldots X_n]$, s sklada umakne skladovni niz $[X_1]_{\pi_1}[X_2]_{\pi_2} \ldots [X_n]_{\pi_n}$ in na sklad potisne

$$[\gamma A]_{[A\longrightarrow X_1X_2\ldots X_n]\,\pi_1\,\pi_2\,\ldots\,\pi_n}.$$

Primer 5: Vzemimo gramatiko G_5 s produkcijami $[S \longrightarrow AB]$, $[A \longrightarrow a]$, $[A \longrightarrow Aa]$, $[B \longrightarrow b]$ in $[B \longrightarrow bB]$. Sintaksna analiza niza $aabb \in L(G_5)$ s Schmeiser-Barnardovim LR(1) analizatorjem je prikazana v tabeli 1. □

LR analiza deluje po strategiji od-spodaj-navzgor, zato je zadnja prevedba, ki jo opravi LR analizator,

SKLAD	VHOD	AKCIJA
$\$\,[\varepsilon]$	$aabb\,\$$	pomik a ε
$\$\,[a]_{\varepsilon}$	$abb\,\$$	prevedba $[A \longrightarrow a]$
$\$\,[A]_{[A\longrightarrow a]}$	$abb\,\$$	pomik a
$\$\,[A]_{[A\longrightarrow a]}\,[Aa]_{\varepsilon}$	$bb\,\$$	prevedba $[A \longrightarrow Aa]$
$\$\,[A]_{[A\longrightarrow Aa][A\longrightarrow a]}$	$bb\,\$$	pomik b
$\$\,[A]_{[A\longrightarrow Aa][A\longrightarrow a]}\,[Ab]_{\varepsilon}$	$b\,\$$	pomik b
$\$\,[A]_{[A\longrightarrow Aa][A\longrightarrow a]}\,[Ab]_{\varepsilon}\,[Abb]_{\varepsilon}$	$\$$	prevedba $[B \longrightarrow b]$
$\$\,[A]_{[A\longrightarrow Aa][A\longrightarrow a]}\,[Ab]_{\varepsilon}\,[AbB]_{[B\longrightarrow b]}$	$\$$	prevedba $[B \longrightarrow bB]$
$\$\,[A]_{[A\longrightarrow Aa][A\longrightarrow a]}\,[AB]_{[B\longrightarrow bB][B\longrightarrow b]}$	$\$$	prevedba $[S \longrightarrow AB]$
$\$\,[S]_{[S\longrightarrow AB][A\longrightarrow Aa][A\longrightarrow a][B\longrightarrow bB][B\longrightarrow b]}$	$\$$	izpis $[S \longrightarrow AB][A \longrightarrow Aa][A \longrightarrow a][B \longrightarrow bB][B \longrightarrow b]$

Tabela 1: Sintaksna analiza vhodnega niza $aabb$ po gramatiki G_5 s Schmeiser-Barnardovim LR(1) analizatorjem.

vedno po produkciji, ki razvija simbol v korenu drevesa izpeljav. Prva produkcija leve sledi izpeljave celotnega vhodnega niza je zato znana šele povsem na koncu analize, iz česar sledi, da Schmeiser-Barnardov analizator sploh ne izpisuje produkcij leve sledi med analizo in zato ni sposoben izvajati semantičnih pravil med samo analizo.

Kljub temu, da je tak analizator primeren za vse LR gramatike in je neobčutljiv za levo rekurzijo, o optimalnosti seveda sploh ni mogoče govoriti.

4.3 Leva LR *sintaksna analiza*

Ker Schmeiser-Barnardov analizator ne more izpisati produkcij leve sledi med potekom analize, je bila leta 2005 definirana leva LR analiza [4].

Pot v nedeterminističnem LR(k) stroju, po kateri ta sprejme veljavni prefiks γ in se konča z opisom oblike $[A \to \alpha \bullet \beta, x]$ pri $z \in \mathrm{FIRST}_k(\beta x)$, se imenuje $\langle\gamma, z\rangle$-*pot*. Pri znanem prefiksu u, ki se razvije iz veljavnega prefixa γ, in vsebini vhodnega okna z, vsaka $\langle\gamma, z\rangle$-pot določa neko izpeljavo

$$S \Longrightarrow^{\pi_u}_{\mathrm{lm}} u\delta \text{ pri } z \in \mathrm{FIRST}_k(\delta).$$

Vse leve sledi π_u, ki jih pri znanem prefiksu u določajo $\langle\gamma, z\rangle$-poti, tvorijo množico $\Pi_{u,z}$, najdaljši skupni prefiks teh sledi pa je ravno $\pi_{u,z}$ [4], [9].

Levi LR(k) analizator uporablja *avtomat prefiksov levih sledi* [4], s katerim ugotovi, ali za prefiks u in vsebino vhodnega okna z obstaja $\langle\gamma, z\rangle$-pot, ki določa $\pi_{u,z}$ ($\pi_{u,z}$ je lahko namreč prekratka leva sled, ki zato ustreza neki poti za $\gamma' \sqsubset \gamma$). Če taka $\langle\gamma, z\rangle$-pot obstaja, izračuna

- veljavni sufiks δ^{R} in
- prefiks leve sledi $\pi_{u,z}$.

Tega izpiše, a obenem pazi, da že v prejšnjih korakih izpisanega dela prefiksa leve sledi $\pi_{u,z}$ ne izpiše še enkrat. Če taka pot ne obstaja, potem uporabi Schmeiser-Barnardov pristop začasnega hranjenja delov leve sledi na skladu.

Primer 6: Leva LR(1) analiza niza $aabb \in G_5$ je prikazana v tabeli 2. Na začetku analize je analizator

SKLAD	VHOD	IZHOD
$\$\,[\varepsilon]$	$aabb\,\$$	$[S \longrightarrow AB]$
$\$\,[a]_{\varepsilon}$	$abb\,\$$	
$\$\,[A]_{[A\longrightarrow a]}$	$abb\,\$$	
$\$\,[A]_{[A\longrightarrow a]}\,[Aa]_{\varepsilon}$	$bb\,\$$	$[A \longrightarrow Aa][A \longrightarrow a]$
$\$\,[A]$	$bb\,\$$	
$\$\,[A]\,[Ab]$	$b\,\$$	$[B \longrightarrow bB]$
$\$\,[A]\,[Ab]\,[Abb]$	$\$$	$[B \longrightarrow b]$
$\$\,[A]\,[Ab]\,[AbB]$	$\$$	
$\$\,[A]\,[AB]$	$\$$	
$\$\,[S]$	$\$$	

Tabela 2: Sintaksna analiza vhodnega niza $aabb$ po gramatiki G_5 z levim LR(1) sintaksnim analizatorjem (akcije so enake kot v tabeli 1).

v stanju $[\varepsilon]$, ki vsebuje veljavne LR(1) opise

$$[S' \to \bullet S, \$], [S \to \bullet AB, \$],$$
$$[A \to \bullet a, b], [A \to \bullet Aa, b],$$
$$[A \to \bullet a, a], [A \to \bullet Aa, a].$$

Vsi ti opisi so aktivni za simbol a v vhodnem oknu, zato lahko analizator napove zgolj razvoj simbola S, ne pa tudi razvoja simbola A, saj ne ve, po kateri poti in s tem po kateri produckiji se ta razvije.. Po pomiku prvega a-ja preide v stanje $[a]$ z opisoma

$$[A \to a\bullet, a] \text{ in } [A \to a\bullet, b].$$

Če bi bil v vhodnem oknu simbol b, bi obstajala enolična pot nazaj do opisa $[S' \to \bullet S, \$] \in [\varepsilon]$. Za simbol a v vhodnem oknu obstajata vsaj dve različni poti (ena preko opisa $[A \to \bullet Aa, a] \in [\varepsilon]$, druga pa ne) in zato vsaj dva različna prefiksa leve sledi vhodnega niza $aabb$.

Ko levi LR(k) analizator potisne prvi a na sklad, izhod vsebuje zgolj $[S \longrightarrow AB]$, čeprav bi (po analogiji s primerom 4) moral vsebovati

$$\pi_{a,a} = [S \longrightarrow AB][A \longrightarrow Aa].$$

Leva LR analiza torej ni optimalna. □

4.4 Sintaksna analiza vrste ALL(*)

Sintaksna analiza vrste ALL(*) se od predhodno obravnavanih metod loči po tem, da ne uporablja vhodnega okna vnaprej določene dolžine. A ker v osnovi opravi sintaksno analizo vhodnega niza po strategiji od-zgoraj-navzdol, predvsem pa zato, ker na njej temelji trenutno izredno popularno orodje ANTLR v4.* [14], si seveda brez dvoma zasluži vsaj kratko obravnavo.

Osnovno razlago sintaksne analize vrste ALL(*) naj si bralec ogleda v delu [14], njeno konkretno implementacijo v orodju ANTLR pa v delu [6]. Na tem mestu bomo z dvema primeroma predstavili le za obravnavo izpisa leve sledi pomembni lastnosti ALL(*) analize.

Primer 7: Vzemimo levo rekurzivno gramatiko G_7 s produkcijami $[S \longrightarrow A]$, $[A \longrightarrow aBbC]$, $[B \longrightarrow Bb]$, $[B \longrightarrow b]$, $[C \longrightarrow Cc]$ and $[C \longrightarrow c]$. Za vhodno besedo $abbbbccc \in L(G_7)$ sintaksna analiza vrste ALL(*) izpiše sled izpeljave

$$[S \longrightarrow A][A \longrightarrow aBbC]$$
$$[B \longrightarrow b][B \longrightarrow Bb][B \longrightarrow Bb]$$
$$[C \longrightarrow c][C \longrightarrow Cc][C \longrightarrow Cc],$$

kar seveda ni leva sled, ustreza pa rezultatu levokotne sintaksne analize vrste XLC(1) ali LAXLC(1) [15], [9].

Primer 8: Vzemimo spet gramatiko G_4 iz primera 4. Orodje ANTLR v4.* zgradi sintaksni analizator za to gramatiko, gramatika $\bar{G}_4$ pa ni primerna za ALL(*) analizo. Pri pretvorbi se namreč produkcija $p_2 = [A \longrightarrow Aa]$ spremeni v produkcijo

$$\bar{p}_2 = [A \longrightarrow p_{2,0}\, A\, p_{2,1}\, a\, p_{2,2}],$$

ANTLR v4.* pa ne more izdelati analizatorja zaradi vstavljene produkcije $[p_{2,0} \longrightarrow \varepsilon]$. Oziroma, v sintaksnem analizatorju vrste ALL(*) za gramatiko G_4 ni mogoče vstaviti semantičnega pravila takoj na začetek desne strani produkcije p_2. □

S primeroma seveda nočemo napadati orodja ANTLR v4.*. Vendar je dejstvo, da je vrstni red izračuna semantičnih pravil manj pregleden kot pri LL ali LR analizi. Pri zapletenejših gramatikah, kjer se prepletata leva in desna rekurzija, je to lahko za pisca prevajalnika precej neprijetno.

5 SKLEP

V tem članku je predstavljen kriterij za optimalni izpis leve sledi izpeljave vhodnega niza. Ta upošteva zahteve, ki nastanejo pri izračunu semantičnih pravil v okviru sintaksno usmerjenega prevajanja.

Po pričakovanjih sintaksna analiza vrste LL izpolni ta kriterij (z njo vred pa tudi SLL analiza kot poenostavljena različica LL analize). Preostale obravnavane vrste sintaksne analize, ki so preimerne za analizo na podlagi širšega razreda gramatik, se optimalnosti le bolj ali manj približajo. Schmeiser-Barnardov analizator se optimalnosti niti ne približa. Leva LR analiza se izkaže precej bolje. Levo sled izračuna s prepletanjem postopka sintaksne analize in izračun semantičnih pravil, zatakne pa se pri levo rekurzivnih produkcijah, ko se do izhoda iz leve rekurzije izpis produkcij in s tem izračun semantičnih pravil ustavi.

Algoritem ALL(*), ki ga uporablja trenutno najpopularnejše orodje za izdelavo sintaksnih analizatorjev, ravno tako ne doseže zahtev kriterija. Glavni problem je precej preprost: ALL(*) analizator v primeru levo rekurzivnih produkcij ne izračuna leve sledi izpeljave (sicer pa razmeroma sproti izpisuje produkcije). Dolžina vhodnega okna, ki ni vnaprej fiksno določena, ni problematična.

Na koncu ostaja vprašanje, ali se da levo LR(k) analizo spremeniti ali dopolniti do te mere, da bi dosegla optimalnost.

LITERATURA

[1] A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, *Compilers: Principles, Techniques, & Tools*, Addison-Wesley, 2006.

[2] Z. Fülöp, H. Vogler, *Syntax-Directed Semantics: Formal Models Based on Tree Transducers*, Springer-Verlag, 1998.

[3] J. P. Schmeiser, D. T. Barnard, "Producing a top-down parse order with bottom-up parsing", *Information Processing Letters*, vol. 54, no. 6, pp. 323–326, 1995.

[4] B. Slivnik, "Producing the left parse during bottom-up parsing", *Information Processing Letters*, vol. 96, no. 6, pp. 220–224, 2005.

[5] E. Scott and A. Johnstone, "GLL Parsing", *Electronic Notes in Theoretical Computer Science*, Springer-Verlag, vol. 253, no. 7, pp. 177–189, 2010.

[6] T. Parr, K. Fischer, "LL(*): The Foundation of the ANTLR Parser Generator", *ACM SIGPLAN Notices*, vol. 46, no. 6, 425–436, 2011.

[7] T. Parr, S. Harwell, K. Fischer, "Adaptive LL(*) parsing: The power of dynamic analysis", *Proc. of the 2014 ACM SIGPLAN Int. Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'14)*, pp. 579–598, 2014.

[8] B. Slivnik, "LLLR Parsing: a Combination of LL and LR parsing", *Proceedings of the 5th Symposium on Languages, Applications and Technologies (SLATE'16)*, 2016.

[9] B. Slivnik, "On different LL and LR parsers used in LLLR parsing", *Computer Languages, Systems & Structures*, vol. 50, no. C, pp. 108–126, 2017.

[10] S. Sippu and E. Soisalon-Soininen, *Parsing Theory, Volume I: Languages and Parsing*, EATCS Monographs on Theoretical Computer Science, vol. 15, Springer-Verlag, 1988.

[11] S. Sippu and E. Soisalon-Soininen, *Parsing Theory, Volume II:* LR(k) *and* LL(k) *Parsing*, EATCS Monographs on Theoretical Computer Science, vol. 20, Springer-Verlag, 1990.

[12] J. Janoušek, B. Melichar, "The output-store formal translator directed by LR parsing", *Lecture Notes in Computer Science*, Springer-Verlag, vol. 1338, pp. 432–439, 1997.

[13] P. M. Lewis II, R. E. Stearns, Syntax directed trasnduction, *Proc. of the 7th Annual Symp. on Switching and Automata Theory (SWAT'66)*, pp. 31–36, 1966.

[14] T. Parr, *The Definitive ANTLR 4 Reference*, 2. izdaja, Pragmatic Bookshelf, 2013.

[15] N. R. Horpsool, Recursive Ascent-Descent Parsing, *Journal of Computer Languages*, vol. 18, no. 1, pp. 1–16, 1993.

Boštjan Slivnik je docent za področje računalništva in informatike na Univerzi v Ljubljani, kjer je leta 2003 doktoriral. Njegovo osnovno raziskovalno področje so prevajalniki s poudarkom na sintaksni analizi, poleg tega pa se ukvarja tudi s porazdeljenimi sistemi in algoritmi. Od leta 1996 je član združenja ACM.