

Prevodenje programskih jezika – beleške sa predavanja

Sintaksna analiza

Milan Banković

* Matematički fakultet,
Univerzitet u Beogradu

Jesenji semestar 2024/25.

Pregled

- 1** Uvod
- 2** Sintaksna analiza (parsiranje) naniže
- 3** Sintaksna analiza (parsiranje) naviše

Sintaksna analiza

Šta je cilj sintaksne analize?

Cilj sintaksne analize je da za datu reč $w \in \Sigma^*$ i datu gramatiku G konstruiše izvođenje reči w u gramatici G , ako postoji.

A šta je sa sintaksnim stablom?

- Kada odredimo izvođenje reči w u gramatici G , iz njega direktno možemo konstruisati i stablo izvođenja
- Iz stabla izvođenja možemo dobiti i stablo apstraktne sintakse

Otuda je najteži deo sintaksne analize upravo određivanje izvođenja date reči u gramatici. U ovoj lekciji prikazujemo osnovne pristupe ovom problemu.

Napomena

Sintaksna analiza se često naziva i [parsiranje](#) (engl. [parsing](#)). I mi ćemo u nastavku često koristiti ovaj izraz.

Levo parsiranje

Definicija 1

Metod parsiranja kod koga je cilj konstrukcija najlevljeg izvođenja date reči w u gramatici G naziva se levo parsiranje.

Levo parsiranje

Postupak levog parsiranja

- U svakom trenutku imamo tekuću rečeničnu formu oblika $uA\beta$ ($u \in \Sigma^*$, $\beta \in (\Sigma \cup N)^*$, tj. A je prvi neterminal sa leva)
 - na početku je to početna rečenična forma S' (početni simbol proširene gramatike: $S' \rightarrow S \dashv$)
- Ako je $u \neq \epsilon$, tada je potrebno da i trenutni ulaz bude oblika uv (u suprotnom imamo sintaksnu grešku)
 - čitamo u sa ulaza i brišemo u sa početka rečenične forme, čime tekuća rečenična forma postaje $A\beta$
 - ovaj postupak predstavlja **sinhronizaciju** ulaza i tekuće rečenične forme
- Ako je $u = \epsilon$, tj. ako je tekuća rečenična forma oblika $A\beta$, tada primenjujemo neko A -pravilo gramatike, npr. $A \rightarrow \gamma$, čime se tekuća rečenična forma transformiše u $\gamma\beta$
 - ovaj postupak predstavlja **ekspanziju** neterminala na osnovu pravila gramatike
- Postupak se završava ili kada nastupi sintaksna greška, ili kada se ulaz potpuno pročita (a rečenična forma potpuno sinhronizuje sa ulazom)

Primer

Primer

Neka je data gramatika izraza:

$$\begin{array}{lcl} E & \longrightarrow & E + T \\ & | & T \\ T & \longrightarrow & T * F \\ & | & F \\ F & \longrightarrow & (E) \\ & | & a \end{array}$$

i reč $a + a * a$. Proširimo ovu gramatiku pravilom $S \rightarrow E \dashv$, gde je S novi početni simbol. Najlevije izvođenje za ulaz $a + a * a \dashv$ je: $S \Rightarrow E \dashv \Rightarrow E + T \dashv \Rightarrow T + T \dashv \Rightarrow F + T \dashv \Rightarrow a + T \dashv \Rightarrow a + T * F \dashv \Rightarrow a + F * F \dashv \Rightarrow a + a * F \dashv \Rightarrow a + a * a \dashv$. Ovo izvođenje možemo rekonstruisati sa leva na desno na sledeći način:

| Tekuća rečenična forma | Tekući ulaz | Primjeno pravilo |
|------------------------|--------------------|---------------------------|
| S | $a + a * a \dashv$ | početna forma |
| $E \dashv$ | $a + a * a \dashv$ | $S \rightarrow E \dashv$ |
| $E + T \dashv$ | $a + a * a \dashv$ | $E \longrightarrow E + T$ |
| $T + T \dashv$ | $a + a * a \dashv$ | $E \longrightarrow T$ |
| $F + T \dashv$ | $a + a * a \dashv$ | $T \longrightarrow F$ |
| $a + T \dashv$ | $a + a * a \dashv$ | $F \longrightarrow a$ |
| $T \dashv$ | $a * a \dashv$ | sinhronizacija |
| $T * F \dashv$ | $a * a \dashv$ | $T \longrightarrow T * F$ |
| $F * F \dashv$ | $a * a \dashv$ | $T \longrightarrow F$ |
| $a * F \dashv$ | $a * a \dashv$ | $F \longrightarrow a$ |
| $F \dashv$ | $a \dashv$ | sinhronizacija |
| $a \dashv$ | $a \dashv$ | $F \longrightarrow a$ |
| ϵ | ϵ | sinhronizacija |

Ovim je parsiranje uspešno rekonstruisano. Redosled primene pravila u najlevijem izvođenju (sa leva na desno) dat je u gornjoj tabeli (odozgo na dole).

Levo parsiranje i potisni automati

Levo parsiranje i potisni automati

Opisani postupak levog parsiranja se može simulirati pomoću **standardnog potisnog automata** pridruženog gramatici G :

- Tekuća rečenična forma se čuva na steku (najlevlji simbol rečenične forme je na vrhu steka)
- Početni simbol steka je početni simbol proširene gramatike
- Sinhronizacija ulaza sa prefiksom rečenične forme vrši se prelazima $(q_0, \varepsilon) \in \delta(q_0, a, a)$, za $a \in \Sigma$
- Ekspanzija najlevljeg neterminala A u tekućoj rečeničnoj formi po pravilu $A \rightarrow \gamma$ vrši se prelazima $(q_0, \gamma) \in \delta(q_0, \varepsilon, A)$

Automat prepoznaje jezik praznim stekom. Redosled primenjenih pravila u najlevljem izvođenju se može rekonstruisati redosledom ekspanzionih prelaza u izračunavanju automata.

Primer

Primer

Vratimo se ponovo na proširenu gramatiku izraza iz prethodnog primera. Standardni automat za ovu gramatiku ima sledeće prelaze:

$$\begin{aligned}
 \delta(q_0, \varepsilon, S) &= \{(q_0, E \dashv)\} \\
 \delta(q_0, \varepsilon, E) &= \{(q_0, E + T), (q_0, T)\} \\
 \delta(q_0, \varepsilon, T) &= \{(q_0, T * F), (q_0, F)\} \\
 \delta(q_0, \varepsilon, F) &= \{(q_0, (E)), (q_0, a)\} \\
 \delta(q_0, a, a) &= \{(q_0, \varepsilon)\} \\
 \delta(q_0, (, ()) &= \{(q_0, \varepsilon)\} \\
 \delta(q_0, (),)) &= \{(q_0, \varepsilon)\} \\
 \delta(q_0, +, +) &= \{(q_0, \varepsilon)\} \\
 \delta(q_0, *, *) &= \{(q_0, \varepsilon)\} \\
 \delta(q_0, \dashv, \dashv) &= \{(q_0, \varepsilon)\}
 \end{aligned}$$

Za reč $a + a * a$ imamo izračunavanje u automatu:

$$\begin{aligned}
 (q_0, a + a * a \dashv, S) &\vdash (q_0, a + a * a \dashv, E \dashv) \vdash (q_0, a + a * a \dashv, E + T \dashv) \vdash (q_0, a + a * a \dashv, T + T \dashv) \vdash (q_0, a + a * a \dashv, F + T \dashv) \vdash (q_0, a + a * a \dashv, a + T \dashv) \vdash (q_0, + a * a \dashv, + T \dashv) \vdash (q_0, a * a \dashv, T \dashv) \vdash (q_0, a * a \dashv, T * F \dashv) \vdash (q_0, a * a \dashv, F * F \dashv) \vdash (q_0, a * a \dashv, a * F \dashv) \vdash (q_0, * a \dashv, * F \dashv) \vdash (q_0, a \dashv, F \dashv) \vdash (q_0, a \dashv, a \dashv) \vdash (q_0, \dashv, \dashv) \vdash (q_0, \varepsilon, \varepsilon).
 \end{aligned}$$

Levo parsiranje

Važna primedba

S obzirom da se kod levog parsiranja najlevlje izvođenje rekonstruiše sa leva na desno, tome odgovara konstrukcija stabla izvođenja od korena ka listovima. Otuda se levo parsiranje često naziva i sintaksna analiza naniže (engl. top down parsing).

Desno parsiranje

Definicija 2

*Metod parsiranja kod koga je cilj konstrukcija najdešnjeg izvođenja date reči w u gramatici G naziva se **desno parsiranje**.*

Desno parsiranje

Postupak desnog parsiranja

- U svakom trenutku nam je poznat prefiks $\alpha \in (\Sigma \cup N)^*$ tekuće rečenične forme koja je oblika αv , a na ulazu imamo nepročitani deo v polazne reči $w = uv \in \Sigma^*$.
 - drugim rečima, tekuća rečenična forma αv je podeljena na deo koji je **otkriven** i deo koji je ostao na ulazu
 - na početku imamo završnu rečeničnu formu w – njen otkriveni prefiks je $\alpha = \varepsilon$, a na ulazu je w
- ukoliko je $\alpha = \alpha' \gamma$, gde je γ desna strana pravila $A \rightarrow \gamma$, tada se može primeniti **redukcija**: γ se zamenjuje sa A , čime otkriveni prefiks postaje $\alpha' A$, dok na ulazu ostaje v
 - primenom u pravom trenutku, redukcijom se postiže rekonstrukcija prethodne rečenične forme $\alpha' Av$ u najdešnjem izvođenju niske w , tj. rekonstruiše se korak $\alpha' Av \xrightarrow{nd} \alpha' \gamma v$ izvođenja
 - niska γ naziva se **ručka** rečenične forme $\alpha' \gamma v$
 - uzastopnim redukcijama se najdešnje izvođenje rekonstruiše **sa desna u levo**, sve dok se ne dostigne početna rečenična forma S
- ukoliko je $v = av'$, tada možemo primeniti **prebacivanje**: a prebacujemo na kraj otkrivenog prefiksa rečenične forme, tj. on postaje αa , a na ulazu ostaje v'
 - da bi mogla da se izvrši prava redukcija, potrebno je da dovoljno dugačak prefiks tekuće rečenične forme bude otkriven
 - upravo to se postiže čitanjem ulaza i prebacivanjem pročitanih simbola u otkriveni deo rečenične forme
- postupak se završava kada se ulaz potpuno pročita, a rečenična forma (koja je sada potpuno otkrivena) bude redukovana u početni simbol S gramatike

Primer

Primer

Neka je data gramatika izraza:

$$\begin{array}{lcl} E & \longrightarrow & E + T \\ & | & T \\ T & \longrightarrow & T * F \\ & | & F \\ F & \longrightarrow & (E) \\ & | & a \end{array}$$

i reč $a + a * a$. Proširimo ovu gramatiku pravilom $S \rightarrow E \dashv$, gde je S novi početni simbol. Najdešnje izvođenje za ulaz $a + a * a \dashv$ je:

$S \Rightarrow E \dashv \Rightarrow E + T \dashv \Rightarrow E + T * F \dashv \Rightarrow E + T * a \dashv \Rightarrow E + F * a \dashv \Rightarrow E + a * a \dashv \Rightarrow T + a * a \dashv \Rightarrow F + a * a \dashv \Rightarrow a + a * a \dashv$.

Ovo izvođenje možemo rekonstruisati sa desna na levo na sledeći način:

| Tekući otkriveni prefiks | Tekući ulaz | Primenjeno pravilo (unazad) |
|--------------------------|--------------------|------------------------------------|
| ϵ | $a + a * a \dashv$ | početna forma |
| a | $+a * a \dashv$ | prebacivanje |
| F | $+a * a \dashv$ | redukcija $F \rightarrow a$ |
| T | $+a * a \dashv$ | redukcija $T \rightarrow F$ |
| E | $+a * a \dashv$ | redukcija $E \rightarrow T$ |
| $E +$ | $a * a \dashv$ | prebacivanje |
| $E + a$ | $*a \dashv$ | prebacivanje |
| $E + F$ | $*a \dashv$ | redukcija $F \rightarrow a$ |
| $E + T$ | $*a \dashv$ | redukcija $T \rightarrow F$ |
| $E + T *$ | $a \dashv$ | prebacivanje |
| $E + T * a$ | \dashv | prebacivanje |
| $E + T * F$ | \dashv | redukcija $F \rightarrow a$ |
| $E + T$ | \dashv | redukcija $T \rightarrow T * F$ |
| E | \dashv | redukcija $E \rightarrow E + T$ |
| $E \dashv$ | ϵ | prebacivanje |
| S | ϵ | redukcija $S \rightarrow E \dashv$ |

Ovim je parsiranje uspešno rekonstruisano. Redosled primene pravila u najdešnjem izvođenju (sa desna na levo) dat je u gornjoj tabeli (odozgo na dole).

Desno parsiranje i potisni automati

Desno parsiranje i potisni automati

Opisani postupak desnog parsiranja se može simulirati pomoću standardnog proširenog potisnog automata pridruženog gramatici G :

- otkriveni prefiks se čuva na steku automata (na vrhu steka je najdešnji simbol ovog prefiksa)
- početni simbol steka je specijalni simbol \perp (marker kraja steka)
- prebacivanje simbola a sa ulaza na stek vrši se prelazom $(q_0, a) \in \delta(q_0, a, \varepsilon)$
- redukcija po pravilu $A \longrightarrow \gamma$ vrši se prelazom $(q_0, A) \in \delta(q_0, \varepsilon, \gamma)$
- dodatni prelaz $(q_0, \varepsilon) \in \delta(q_0, \varepsilon, \perp S)$ omogućava pražnjenje steka nakon poslednje redukcije

Automat prepozna jezik praznim stekom. Redosled primenjenih pravila u najdešnjem izvođenju se može rekonstruisati **unazad** redosledom redukcionih prelaza u izračunavanju automata.

Primer

Primer

Vratimo se ponovo na proširenu gramatiku izraza iz prethodnog primera. Standardni prošireni automat za ovu gramatiku ima sledeće prelaze:

$$\begin{aligned}
 \delta(q_0, a, \varepsilon) &= \{(q_0, a)\} \\
 \delta(q_0, (, \varepsilon) &= \{(q_0, ())\} \\
 \delta(q_0,), \varepsilon) &= \{(q_0,))\} \\
 \delta(q_0, +, \varepsilon) &= \{(q_0, +)\} \\
 \delta(q_0, *, \varepsilon) &= \{(q_0, *)\} \\
 \delta(q_0, \neg, \varepsilon) &= \{(q_0, \neg)\} \\
 \delta(q_0, \varepsilon, E \dashv) &= \{(q_0, S)\} \\
 \delta(q_0, \varepsilon, E + T) &= \{(q_0, E)\} \\
 \delta(q_0, \varepsilon, T) &= \{(q_0, E)\} \\
 \delta(q_0, \varepsilon, T * F) &= \{(q_0, T)\} \\
 \delta(q_0, \varepsilon, F) &= \{(q_0, T)\} \\
 \delta(q_0, \varepsilon, (E)) &= \{(q_0, F)\} \\
 \delta(q_0, \varepsilon, a) &= \{(q_0, F)\} \\
 \delta(q_0, \varepsilon, \perp S) &= \{(q_0, \varepsilon)\}
 \end{aligned}$$

Za reč $a + a * a$ imamo izračunavanje u automatu: $(q_0, a + a * a \dashv, \perp) \vdash (q_0, + a * a \dashv, \perp a) \vdash (q_0, + a * a \dashv, \perp F) \vdash (q_0, + a * a \dashv, \perp T) \vdash (q_0, + a * a \dashv, \perp E) \vdash (q_0, a * a \dashv, \perp E +) \vdash (q_0, * a \dashv, \perp E + a) \vdash (q_0, * a \dashv, \perp E + F) \vdash (q_0, * a \dashv, \perp E + T) \vdash (q_0, a \dashv, \perp E + T *) \vdash (q_0, \dashv, \perp E + T * a) \vdash (q_0, \dashv, \perp E + T * F) \vdash (q_0, \dashv, \perp E + T) \vdash (q_0, \dashv, \perp E) \vdash (q_0, \varepsilon, \perp E \dashv) \vdash (q_0, \varepsilon, \perp S) \vdash (q_0, \varepsilon, \varepsilon)$.

Desno parsiranje

Važna primedba

S obzirom da se kod desnog parsiranja najdešnje izvođenje rekonstruiše sa desna na levo, tome odgovara konstrukcija stabla izvođenja od listova ka korenu. Otuda se desno parsiranje često naziva i sintaksna analiza naviše (engl. bottom up parsing).

Determinizam

Kako postići determinizam?

- Prethodni postupci parsiranja (i pridruženi automati) su po pravilu nedeterministički
 - Kod analize naniže (levog parsiranja) problem je koje pravilo primeniti na neterminal na vrhu steka
 - Kod analize naviše (desnog parsiranja) često je moguće primeniti i redukciju i prebacivanje, ili dve različite redukcije (po različitim pravilima)
- Ove više značnosti se, na žalost, ne mogu uvek otkloniti, s obzirom da znamo da postoje nedeterministički kontekstno slobodni jezici
- Za odredjene klase gramatika (i jezika), nedeterminizam se može ukloniti korišćenjem preduvidnih simbola
 - drugim rečima, simboli koji slede na ulazu nam mogu pomoći da odlučimo koju ekspanziju (kod analize naniže), odnosno, koju redukciju ili prebacivanje (kod analize naviše) da primenimo
- Ključna pitanja:
 - kako utvrditi da li data gramatika omogućava determinističko parsiranje na ovaj način?
 - kako odrediti **tablicu prelaska** koja na osnovu preduvidnog simbola i simbola na vrhu steka određuje koji prelaz automata treba primeniti?
- Na ova pitanja odgovaramo u nastavku

Pregled

1 Uvod

2 Sintaksna analiza (parsiranje) naniže

3 Sintaksna analiza (parsiranje) naviše

LL(1) gramatike

Pitanje na koje tražimo odgovor:

Koje uslove treba da zadovoljava gramatika da bi se parsiranje naniže moglo obaviti na deterministički način na osnovu jednog preduvidnog simbola?

- Nedeterminizam kod parsiranja naniže je posledica postojanja više pravila gramatike za isti neterminal A koji se nalazi na vrhu steka
- Potrebno je za svako pravilo $A \rightarrow \gamma$ odrediti skup preduvidnih simbola iz Σ za koje je u izvođenju nalevo potrebno primeniti baš to pravilo

Definicija 3

Skup izbora za pravilo $A \rightarrow \gamma$ (u oznaci $C(A \rightarrow \gamma)$) je skup terminala $a \in \Sigma \cup \{\vdash\}$ takvih da postoji najlevije izvođenje:

$$S \xrightarrow{nl,*} uA\beta \xrightarrow{} u\gamma\beta \xrightarrow{nl,*} uav \in \Sigma^*$$

pri čemu je $u, v \in \Sigma^$, $\alpha, \beta \in (\Sigma \cup N)^*$.*

Primedba

Da bi parsiranje bilo determinističko, za svaka dva različita A -pravila ovi skupovi moraju biti disjunktni (za svako $A \in N$).

LL(1) gramatike

Definicija 4

Gramatika $G = (\Sigma, N, S, P)$ je $\text{LL}(1)$ akko za svaka dva pravila $A \rightarrow \gamma_1$ i $A \rightarrow \gamma_2$ sa istom levom stranom važi da je $C(A \rightarrow \gamma_1) \cap C(A \rightarrow \gamma_2) = \emptyset$.

Primedba

Dakle, $\text{LL}(1)$ gramatike su one gramatike koje dopuštaju determinističko parsiranje naniže, koristeći jedan preduvidni simbol sa ulaza.

Definicija 5

Jezik L je $\text{LL}(1)$ jezik ako postoji $\text{LL}(1)$ gramatika G takva da je $L = L(G)$.

LL(1) gramatike

Odlučivost

- Pitanje da li je gramatika $LL(1)$ je **odlučivo** i svodi se na izračunavanje skupova izbora (u nastavku ove lekcije)
- Pitanje da li je jezik $LL(1)$ je **neodlučivo** u opštem slučaju
 - ovo znači da ne postoji opšti postupak za transformaciju proizvoljne gramatike u $LL(1)$ gramatiku i/ili za ispitivanje da li je to uopšte moguće
- U praksi se mnoge gramatike koje nisu $LL(1)$ mogu transformisati u ekvivalentne $LL(1)$ gramatike primenom nekih uobičajenih transformacija:
 - eliminacija leve rekurzije, leva faktorizacija, ...
 - ovo, naravno, ne prolazi uvek, zato uvek treba nakon tih transformacija proveriti da li je dobijena gramatika $LL(1)$
- Postoje jezici koji nisu $LL(1)$, a jesu deterministički
 - Ovo znači da parsiranje naniže uz pomoć jednog preduvidnog simbola nije metod koji je dovoljno moćan da pokrije sve determinističke jezike

Skup $Prvi(\alpha)$

Definicija 6

Neka je data gramatika $G = (\Sigma, N, S, P)$ i neka je $\alpha \in (\Sigma \cup N)^*$. Tada je:

$$Prvi(\alpha) = \{a \in \Sigma \mid \alpha \xrightarrow{*} aw, w \in \Sigma^*\}$$

Algoritam

Skup $Prvi(\alpha)$ određujemo na sledeći način:

- ako je $\alpha = a \in \Sigma$, tada je $Prvi(\alpha) = \{a\}$
- ako je $\alpha = A \in N$, tada:
 - formiramo graf čiji su čvorovi iz $\Sigma \cup N$
 - grana od $A \in N$ do $X \in (\Sigma \cup N)$ će postojati akko postoji pravilo $A \longrightarrow \beta X \gamma$, pri čemu je β anulirajuća niska (tj. $\beta \xrightarrow{*} \varepsilon$, odnosno, β je ili ε ili se sastoji samo iz anulirajućih neterminala)
 - ako u grafu postoji put od $A \in N$ do $a \in \Sigma$, tada $a \in Prvi(A)$
- ako je $\alpha = X_1 X_2 \dots X_n$ ($X_i \in (\Sigma \cup N)$ za $i = 1, 2, \dots, n$), i ako je X_k prvi simbol u nizu koji je ili terminal, ili neanulirajući neterminal, tada je

$$Prvi(\alpha) = Prvi(X_1) \cup Prvi(X_2) \cup \dots \cup Prvi(X_k)$$

Primer

Primer

Neka je data gramatika $S \rightarrow aSb \mid \varepsilon$. Formiramo graf:

$S \dashrightarrow a$

Grana postoji od S do a zato što imamo pravilo $S \rightarrow aSb$. Kako a nije anulirajući, to je i jedina grana iz S . Sada važi da je $Prvi(S) = \{a\}$, jer je a jedini terminal za koji postoji put u grafu od S do a . Dalje, $Prvi(aSb) = \{a\}$, jer je a terminal. Sa druge strane, $Prvi(Sb) = \{a, b\}$, jer je $Prvi(S) = \{a\}$, a kako je S anulirajući neterminal, tada je $Prvi(Sb) = Prvi(S) \cup Prvi(b) = \{a, b\}$.

Skup $Sledi(A)$

Definicija 7

Neka je data gramatika $G = (\Sigma, N, S, P)$, i neka je $A \in N$. Tada je:

$$Sledi(A) = \{a \in \Sigma \mid S \xrightarrow{*} \alpha A a \gamma\}$$

gde su $\alpha, \gamma \in (\Sigma \cup N)^*$.

Algoritam

- Posmatramo proširenu gramatiku

$$G' = (\Sigma \cup \{\dashv\}, N \cup \{S'\}, S', P \cup \{S' \longrightarrow S \dashv\})$$

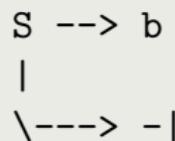
gramatike G

- Konstruišemo graf sa čvorovima iz $\Sigma \cup \{\dashv\} \cup N$:
 - Grana postoji od $A \in N$ do $b \in \Sigma \cup \{\dashv\}$, akko postoji pravilo $X \longrightarrow \alpha A \beta$ i $b \in Prvi(\beta)$
 - Grana postoji od $A \in N$ do $B \in N$, akko imamo pravilo $B \longrightarrow \alpha A \beta$, $\beta \xrightarrow{*} \varepsilon$
- Sada je $a \in \Sigma \cup \{\dashv\}$ u skupu $Sledi(A)$ akko postoji put od A do a u grafu

Primer

Primer

Posmatrajmo ponovo gramatiku $S \rightarrow aSb \mid \varepsilon$ (tj. njenu proširenu gramatiku koja uključuje i pravilo $S' \rightarrow S \dashv$). Formiramo graf:



Grana postoji od S do b jer imamo pravilo $S \rightarrow aSb$, a $b \in \text{Prvi}(b)$. Slično, postoji grana od S do \dashv , jer postoji pravilo $S' \rightarrow S \dashv$ (u proširenoj gramatici), a $\dashv \in \text{Prvi}(\dashv)$. Sada je $\text{Sledi}(S) = \{b, \dashv\}$.

Skup izbora

Teorema 1

$$C(A \rightarrow \gamma) = \begin{cases} Prvi(\gamma), & \text{ako } \gamma \text{ nije anulirajući} \\ Prvi(\gamma) \cup Sledi(A), & \text{ako } \gamma \text{ jeste anulirajući} \end{cases}$$

Dokaz

Da bi postojalo najlevlje izvođenje $S \xrightarrow{nl,*} uA\beta \xrightarrow{} u\gamma\beta \xrightarrow{nl,*} uav$, potrebno je da bude $S \xrightarrow{nl,*} uA\beta$, kao i da $a \in Prvi(\gamma\beta)$. Ako γ nije anulirajuća niska, tada je $Prvi(\gamma\beta) = Prvi(\gamma)$, pa je

$C(A \rightarrow \gamma) = Prvi(\gamma)$. Sa druge strane, ako je γ anulirajuća, tada je $Prvi(\gamma\beta) = Prvi(\gamma) \cup Prvi(\beta)$. Pritom, treba razmatrati sve β za koje je $S \xrightarrow{*} uA\beta$, a unija skupova $Prvi(\beta)$ za takve β je upravo skup $Sledi(A)$. Otuda je $C(A \rightarrow \gamma) = Prvi(\gamma) \cup Sledi(A)$.

Primer

Primer

Vratimo se na gramatiku $S \rightarrow aSb \mid \varepsilon$. Skupovi izbora su:

- $C(S' \rightarrow S \dashv) = Prvi(S \dashv) = Prvi(S) \cup Prvi(\dashv) = \{a, \dashv\}$
- $C(S \rightarrow aSb) = Prvi(aSb) = \{a\}$
- $C(S \rightarrow \varepsilon) = Prvi(\varepsilon) \cup Sledi(S) = \emptyset \cup \{b, \dashv\} = \{b, \dashv\}$

Otuda je ova gramatika $LL(1)$, jer je

$$C(S \rightarrow aSb) \cap C(s \rightarrow \varepsilon) = \emptyset.$$

Tablica prelaska

Tablica prelaska

Tablica prelaska za $LL(1)$ analizu je tablica T u kojoj:

- svakom terminalu gramatike (uključujući i \vdash) odgovara jedna kolona
- svakom neterminalu (uključujući i S') odgovara jedna vrsta
- polja tablice T sadrže pravila gramatike:
 - polje $T[A, a]$ sadrži pravilo $A \longrightarrow \gamma$ akko je $a \in C(A \longrightarrow \gamma)$

Primedba

Ako je gramatika $LL(1)$, tada će svako polje tablice sadržati najviše jedno pravilo.

Primer

Primer

Tablica prelaska za $LL(1)$ analizu za gramatiku $S \rightarrow aSb \mid \varepsilon$ je:

| | a | b | \vdash |
|------|---------------------------|-----------------------------|-----------------------------|
| S' | $S' \rightarrow S \vdash$ | - | $S' \rightarrow S \vdash$ |
| S | $S \rightarrow aSb$ | $S \rightarrow \varepsilon$ | $S \rightarrow \varepsilon$ |

Algoritam $LL(1)$ analize

Algoritam

Inicijalno, na steku imamo početni simbol proširene gramatike S' . U svakom koraku primenjujemo sledeće:

- ako je na vrhu steka neterminal A , a preduvidni karakter je a , tada gledamo polje tablice $T[A, a]$:
 - ako postoji pravilo $A \rightarrow \gamma \in T[A, a]$, tada simbol A na vrhu steka zamenjujemo sa γ ; simbol a ostaje na ulazu
 - ako ne postoji pravilo u $T[A, a]$, prijavljujemo sintaksnu grešku
- ako je na vrhu steka terminal a , tada posmatramo šta je sledeći simbol na ulazu:
 - ako je sledeći simbol na ulazu a , tada uklanjamo a sa ulaza i sa vrha steka
 - ako je sledeći simbol na ulazu različit od a , tada prijavljujemo sintaksnu grešku
- ako se stek isprazni, tada je reč koja je pročitana sa ulaza prepoznata od strane automata

Primer

Primer

Vratimo se opet na gramatiku $S \rightarrow aSb \mid \varepsilon$. Posmatrajmo nisku $aabb$.

Prepoznavanje ove niske se može prikazati sledećom tablicom:

| Stek | Ulaz | Primjenjeno pravilo |
|---------------|---------------|-----------------------------|
| S' | $aabb \dashv$ | - |
| $S \dashv$ | $aabb \dashv$ | $S' \rightarrow S \dashv$ |
| $aSb \dashv$ | $aabb \dashv$ | $S \rightarrow aSb$ |
| $Sb \dashv$ | $abb \dashv$ | sinhronizacija |
| $aSbb \dashv$ | $abb \dashv$ | $S \rightarrow aSb$ |
| $Sbb \dashv$ | $bb \dashv$ | sinhronizacija |
| $bb \dashv$ | $bb \dashv$ | $S \rightarrow \varepsilon$ |
| $b \dashv$ | $b \dashv$ | sinhronizacija |
| \dashv | \dashv | sinhronizacija |
| ε | ε | sinhronizacija |

Primer

Primer

Neka je data gramatika izraza:

$$\begin{array}{lcl} E & \longrightarrow & E + T \\ & | & \\ & T & \\ T & \longrightarrow & T * F \\ & | & \\ & F & \\ F & \longrightarrow & (E) \\ & | & \\ & a & \end{array}$$

i reč $a + a * a$. Proširimo ovu gramatiku pravilom $S \longrightarrow E \dashv$, gde je S novi početni simbol. Ispitajmo da li je ova gramatika $LL(1)$. Odredimo najpre skupove Prvi i Sledi. Konstruišemo graf (primetimo da ni jedan od simbola E , T i F nije anulirajući):

Prvi:

$$\begin{array}{l} E \longrightarrow T \longrightarrow F \longrightarrow (\\ | \\ \backslash \longrightarrow a \end{array}$$

Sledi:

$$\begin{array}{lll} F \longrightarrow & T \longrightarrow & E \longrightarrow + \\ | & | & | \\ \backslash \longrightarrow * & \backslash \longrightarrow) & | \\ & & \backslash \longrightarrow - \end{array}$$

Odavde je $\text{Prvi}(E) = \text{Prvi}(T) = \text{Prvi}(F) = \{((), a)\}$. Dalje je $\text{Sledi}(E) = \{+, \), $\dashv\}$$, $\text{Sledi}(T) = \{+, \), $\dashv, *\}$ i $\text{Sledi}(F) = \{+, \), $\dashv, *\}$. Otuda su skupovi izbora:$$

- $C(S \longrightarrow E \dashv) = \text{Prvi}(E) = \{((), a)\}$
- $C(E \longrightarrow E + T) = \text{Prvi}(E) = \{((), a)\}$, $C(E \longrightarrow T) = \text{Prvi}(T) = \{((), a)\}$
- $C(T \longrightarrow T * F) = \text{Prvi}(T) = \{((), a)\}$, $C(T \longrightarrow F) = \text{Prvi}(F) = \{((), a)\}$
- $C(F \longrightarrow (E)) = \{\}\}$, $C(F \longrightarrow a) = \{a\}$

Kako skupovi izbora za E -pravila (kao i za T -pravila) nisu disjunktni, gramatika nije $LL(1)$.

$LL(1)$ gramatike i leva rekurzija

Teorema 2

Gramatike koje sadrže levu rekurziju (posrednu ili neposrednu) nisu $LL(1)$.

Dokaz

Dokažimo teoremu za neposrednu levu rekurziju. Neka je gramatika neposredno levo rekurzivna po simbolu A , tj. neka postoji bar jedno levo rekurzivno A -pravilo, kao i bar jedno A -pravilo koje nije levo rekurzivno (koje sigurno postoji, ako je gramatika čista). Na primer, imamo $A \rightarrow A\alpha \mid \beta$. Bez ograničenja opštosti, pretpostavimo da je bar jedan od skupova $Prvi(\beta)$ i $Prvi(\alpha)$ neprazan (u suprotnom bismo iz A mogli da izvedemo samo ϵ , pa bismo A mogli da izbacimo iz gramatike). Sada je $Prvi(\beta) \subseteq Prvi(A)$. Ako je $Prvi(\beta) \neq \emptyset$, tada je jasno da skupovi izbora za $A \rightarrow A\alpha$ i $A \rightarrow \beta$ nisu disjunktni. Ako je $Prvi(\beta) = \emptyset$, to je moguće samo ako je β anulirajuća niska. Međutim, tada je i A anulirajući simbol, pa skup izbora za $A \rightarrow A\alpha$ sadrži i simbole iz $Prvi(\alpha)$. Kako skup izbora za $A \rightarrow \beta$ sadrži skup $Sledi(A)$, a $Prvi(\alpha) \subseteq Sledi(A)$, sledi da skupovi izbora za ova dva pravila imaju $Prvi(\alpha)$ u preseku. Kako je ovaj skup po pretpostavci neprazan, sledi da gramatika ne može biti $LL(1)$.

Primer

Primer

Iz prethodne teoreme je jasno zbog čega prethodna gramatika izraza nije bila $LL(1)$. Prvi pokušaj transformacije ove gramatiku u ekvivalentnu $LL(1)$ gramatiku bi podrazumevao eliminaciju leve rekurzije, pri čemu dobijamo sledeću gramatiku:

$$\begin{array}{lcl} E & \longrightarrow & TE' \\ E' & \longrightarrow & +TE' \mid \epsilon \\ T & \longrightarrow & FT' \\ T' & \longrightarrow & *FT' \mid \epsilon \\ F & \longrightarrow & (E) \\ & & \mid a \end{array}$$

Odredimo skupove Prvi i Sledi za neterminale ove gramatike (anulirajući simboli su E' i T'):

Prvi:

$$\begin{array}{ll} E \rightarrow T \rightarrow F \rightarrow (& \rightarrow T' \quad \rightarrow E' \\ | & | \quad | \quad | \\ | & | \quad \backslash \quad | \quad \backslash \\ \backslash \rightarrow a & F \rightarrow T \rightarrow E \rightarrow) \\ E' \rightarrow + & \rightarrow * \quad \backslash \rightarrow + \quad \backslash \rightarrow - \\ T' \rightarrow * & \end{array}$$

Sledi:

Odavde je $\text{Prvi}(E) = \text{Prvi}(T) = \text{Prvi}(F) = \{(, a\}$, $\text{Prvi}(E') = \{+\}$, $\text{Prvi}(T') = \{* \}$, $\text{Sledi}(E) = \text{Sledi}(E') = \{\}, \dashv\}$, $\text{Sledi}(T) = \text{Sledi}(T') = \{+, \), \dashv\}$ i $\text{Sledi}(F) = \{*, +, \), \dashv\}$.

Primer

Primer

(nastavak) Sada imamo sledeće skupove izbora:

- $C(S \rightarrow E \dashv) = Prvi(E) = \{(, a\}$
- $C(E \rightarrow TE') = Prvi(T) = \{(, a\}$
- $C(E' \rightarrow +TE') = \{+\}, C(E' \rightarrow \varepsilon) = Sledi(E') = \{), \dashv\}$
- $C(T \rightarrow FT') = Prvi(F) = \{(, a\}$
- $C(T' \rightarrow *FT') = \{*}, C(T' \rightarrow \varepsilon) = Sledi(T') = \{+,), \dashv\}$
- $C(F \rightarrow (E)) = \{\}, C(F \rightarrow a) = \{a\}$

Odavde sledi da gramatika jeste $LL(1)$. Tablica prelaska za ovu gramatiku izgleda ovako:

| | a | $($ | $)$ | $+$ | $*$ | \dashv |
|------|--------------------------|--------------------------|------------------------------|------------------------------|-----------------------|------------------------------|
| S | $S \rightarrow E \dashv$ | $S \rightarrow E \dashv$ | - | - | - | - |
| E | $E \rightarrow TE'$ | $E \rightarrow TE'$ | - | - | - | - |
| E' | - | - | $E' \rightarrow \varepsilon$ | $E' \rightarrow +TE'$ | - | $E' \rightarrow \varepsilon$ |
| T | $T \rightarrow FT'$ | $T \rightarrow FT'$ | - | - | - | - |
| T' | - | - | $T' \rightarrow \varepsilon$ | $T' \rightarrow \varepsilon$ | $T' \rightarrow *FT'$ | $T' \rightarrow \varepsilon$ |
| F | $F \rightarrow a$ | $F \rightarrow (E)$ | - | - | - | - |

Primer

Primer

(nastavak) Izvođenje niske $a + a * a$ je dato sledećom tabelom:

| Stek | Ulaz | Primjenjeno pravilo |
|-----------------|--------------------|---------------------------|
| S | $a + a * a \dashv$ | - |
| $E \dashv$ | $a + a * a \dashv$ | $S \rightarrow E \dashv$ |
| $TE' \dashv$ | $a + a * a \dashv$ | $E \rightarrow TE'$ |
| $FT'E' \dashv$ | $a + a * a \dashv$ | $T \rightarrow FT'$ |
| $aT'E' \dashv$ | $a + a * a \dashv$ | $F \rightarrow a$ |
| $T'E' \dashv$ | $+a * a \dashv$ | sinhronizacija |
| $E' \dashv$ | $+a * a \dashv$ | $T' \rightarrow \epsilon$ |
| $+TE' \dashv$ | $+a * a \dashv$ | $E' \rightarrow +TE'$ |
| $TE' \dashv$ | $a * a \dashv$ | sinhronizacija |
| $FT'E' \dashv$ | $a * a \dashv$ | $T \rightarrow FT'$ |
| $aT'E' \dashv$ | $a * a \dashv$ | $F \rightarrow a$ |
| $T'E' \dashv$ | $*a \dashv$ | sinhronizacija |
| $*FT'E' \dashv$ | $*a \dashv$ | $T' \rightarrow *FT'$ |
| $FT'E' \dashv$ | $a \dashv$ | sinhronizacija |
| $aT'E' \dashv$ | $a \dashv$ | $F \rightarrow a$ |
| $T'E' \dashv$ | \dashv | sinhronizacija |
| $E' \dashv$ | \dashv | $T' \rightarrow \epsilon$ |
| \dashv | \dashv | $E' \rightarrow \epsilon$ |
| ϵ | ϵ | sinhronizacija |

Primer

Primer

Posmatrajmo gramatiku:

$$\begin{array}{lcl} S & \longrightarrow & (L) \mid a \\ L & \longrightarrow & S \ L \mid S \end{array}$$

Ova gramatika nema levu rekurziju. Ispitajmo da li je $LL(1)$. Lako se vidi da je $Prvi(L) = Prvi(S) = \{(, a\}$, kao i da je $Sledi(L) = \{\}\}$ i $Sledi(S) = Prvi(L) \cup Sledi(L) \cup \{\dashv\} = \{(, a,), \dashv\}$. Skupovi izbora su:

- $C(S' \rightarrow S \dashv) = Prvi(S) = \{(, a\}$
- $C(S \rightarrow (L)) = \{(\}, C(S \rightarrow a) = \{a\}$
- $C(L \rightarrow S \ L) = Prvi(S) = \{(, a\},$
 $C(L \rightarrow S) = Prvi(S) = \{(, a\}$

Gramatika nije $LL(1)$ jer skupovi izbora za L -pravila nisu disjunktni.

$LL(1)$ gramatike i leva faktorisanost

Teorema 3

Gramatika koja je levo faktorisana nije $LL(1)$.

Dokaz

Leva faktorisanost znači da postoje neka dva pravila $A \rightarrow \alpha\beta_1$ i $A \rightarrow \alpha\beta_2$ koja dele zajednički prefiks α . Kao i u prethodnoj teoremi, bez ograničenja opštosti, pretpostavimo da je $Prvi(\alpha) \neq \emptyset$. Jasno je da će $Prvi(\alpha)$ biti sadržan u oba skupa izbora, pa gramatika nije $LL(1)$.

Primer

Primer

Posmatrajmo ponovo gramatiku iz prethodnog primera. Eliminacijom leve faktorisanosti, dobijamo gramatiku:

$$\begin{array}{lcl} S & \longrightarrow & (L) \mid a \\ L & \longrightarrow & S X \\ X & \longrightarrow & L \mid \epsilon \end{array}$$

Sada je $Prvi(S) = Prvi(L) = Prvi(X) = \{a, ()\}$, a $Sledi(L) = Sledi(X) = \{\}\}$, a $Sledi(S) = \{\dashv\} \cup Prvi(X) \cup Sledi(L) = \{\dashv, a, (), \}\}$. Skupovi izbora su:

- $C(S' \longrightarrow S \dashv) = Prvi(S) = \{a, ()\}$
- $C(S \longrightarrow (L)) = \{()\}, C(S \longrightarrow a) = \{a\}$
- $C(L \longrightarrow S X) = Prvi(S) = \{a, ()\}$
- $C(X \longrightarrow L) = Prvi(L) = \{a, ()\}, C(X \longrightarrow \epsilon) = Sledi(X) = \{\}\}$

Sada je gramatika $LL(1)$.

Primer

Primer

(nastavak) Tablica prelaska za prethodni primer je:

| | a | $($ | $)$ | \vdash |
|------|---------------------------|---------------------------|-----------------------------|----------|
| S' | $S' \rightarrow S \vdash$ | $S' \rightarrow S \vdash$ | - | - |
| S | $S \rightarrow a$ | $S \rightarrow (L)$ | - | - |
| L | $L \rightarrow S X$ | $L \rightarrow S X$ | - | - |
| X | $X \rightarrow L$ | $X \rightarrow L$ | $X \rightarrow \varepsilon$ | - |

Primer

Primer

Iz prethodnih primera, može se pogrešno zaključiti da su jedini uzroci problema kod $LL(1)$ analize leva rekurzija i leva faktorisanost, te da se eliminacijom ovih pojava uvek dobija $LL(1)$ gramatika. Pokažimo da to nije tačno. Posmatrajmo gramatiku:

$$\begin{array}{rcl} S & \longrightarrow & aAaa \mid bAba \\ A & \longrightarrow & b \mid \varepsilon \end{array}$$

Ova gramatika nema ni levu rekurziju, ni levu faktorisanost. Da li je $LL(1)$?

Lako se vidi da je $Prvi(S) = \{a, b\}$, $Prvi(A) = \{b\}$, $Sledi(S) = \{\dashv\}$ i

$Sledi(A) = \{a, b\}$. Sada imamo skupove izbora:

- $C(S' \longrightarrow S \dashv) = Prvi(S) = \{a, b\}$
- $C(S \longrightarrow aAaa) = \{a\}$, $C(S \longrightarrow bAba) = \{b\}$
- $C(A \longrightarrow b) = \{b\}$, $C(A \longrightarrow \varepsilon) = Sledi(A) = \{a, b\}$

Odavde sledi da gramatika nije $LL(1)$, jer skupovi izbora za A -pravila nisu disjunktni.

Rekurzivni spust

Rekurzivni spust

- Jedan način implementacije $LL(1)$ parsera je da se simulira rad odgovarajućeg potisnog automata (tj. algoritam sa slajda 29), uz eksplicitnu implementaciju tablice prelaska
- Jednostavniji način je da se koristi rekurzivna implementacija poznata i kao **rekurzivni spust**:
 - za svaki neterminal imamo posebnu funkciju (obično nazvanu kao i sam neterminal)
 - imamo i globalnu promenljivu *token* koja sadrži pročitani preduvidni token
 - parsiranje se započinje čitanjem prvog preduvidnog simbola i pozivom funkcije $S()$ (gde je S početni simbol proširene gramatike)
 - funkcija $A()$ za proizvoljni neterminal A će na osnovu vrednosti promenljive *token* odlučiti koje A pravilo da primeni
 - desna strana odabranog pravila će se obrađivati sa leva na desno:
 - ako je sledeći simbol desne strane terminal, moraće da bude isti kao pročitani preduvid; tada čitamo sledeći preduvid i nastavljamo dalje; u suprotnom prijavljujemo grešku
 - ako je sledeći simbol desne strane neterminal X , pozivaće se funkcija $X()$
 - parsiranje se završava kada se vratimo iz poziva $S()$

Primer

Primer

Implementacija parsera zasnovanog na rekurzivnom spustu za gramatiku $S \rightarrow aSb \mid \varepsilon$ je data u nastavku:

```
int token;
void parse()
{
    token = next_token();
    if(Sp())      // S'
        printf("Success\n");
    else
        printf("Error\n");
}
```

Primer

Primer

```
int Sp()
{
    if(token == a_token || token == eof_token)
    {
        S(); // S
        return token == eof_token; // -
    }
    else // b_token
        return 0;
}
```

Primer

Primer

```
int S()
{
    if(token == a_token)
    {
        token = next_token(); // a
        if(!S())           // S
            return 0; // greska
        if(token != b_token) // b
            return 0; // greska
        token = next_token();
    }
    else if(token == b_token || token == eof_token)
    {
        // epsilon
    }
    return 1;
}
```

Ograničenja analize naniže

Napomene

- Umesto jednog, u teoriji se razmatra i korišćenje više od jednog preduvidnog simbola
- Gramatike koje dopuštaju determinističko parsiranje naniže uz pomoć k preduvidnih simbola nazivaju se $LL(k)$ gramatike
- Jezici koji se mogu opisati $LL(k)$ gramatikama se nazivaju $LL(k)$ jezici
- Važi: $LL(1) \subsetneq LL(2) \subsetneq \dots \subsetneq LL(k) \subsetneq \dots \subsetneq DKS$, gde je DKS klasa svih determinističkih kontekstno slobodnih jezika
- Dakle, analiza naniže nije dovoljno moćna da obezbedi determinističko parsiranje svih determinističkih kontekstno slobodnih jezika!!

Pregled

- 1 Uvod**
- 2 Sintaksna analiza (parsiranje) naniže**
- 3 Sintaksna analiza (parsiranje) naviše**

Ručke

Definicija 8

Neka je dato najdešnje izvođenje: $S \xrightarrow{nd,*} \alpha A w \xrightarrow{nd} \alpha \beta w$, ($w \in \Sigma^*$, $\alpha, \beta \in (N \cup \Sigma)^*$). Tada se reč β naziva **ručka** rečenične forme $\alpha \beta w$.

Napomena

Dakle, ručka je deo tekuće rečenične forme koji je nastao poslednjim primjenjenim pravilom u najdešnjem izvođenju.

Primer

U sledećem najdešnjem izvođenju, ručke su označene crvenom bojom:

$$\begin{aligned} E' \xrightarrow{} & E \dashv \xrightarrow{} E + T \dashv \xrightarrow{} E + T * F \dashv \xrightarrow{} E + T * a \dashv \xrightarrow{} E + F * a \dashv \xrightarrow{} E + a * a \dashv \xrightarrow{} \\ & T + a * a \dashv \xrightarrow{} F + a * a \dashv \xrightarrow{} a + a * a \dashv. \end{aligned}$$

Primer

Slično, za izvođenje: $S' \xrightarrow{} S \dashv \xrightarrow{} aSb \dashv \xrightarrow{} aaSbb \dashv \xrightarrow{} aa\epsilon bb \dashv$

Ključno pitanje

- Kako, idući sa desna u levo u ovim izvođenjima, prepoznavati ručke na deterministički način?

$LR(1)$ gramatike

Definicija 9

Gramatika je $LR(1)$ akko je tokom parsiranja naviše u svakom koraku na osnovu jednog preduvidnog simbola moguće jednoznačno odrediti da li je potrebno izvršiti prebacivanje ili redukciju, kao i po kom pravilu se redukcija vrši.

Napomena

Ova definicija je neformalna, ali dobro izražava suštinu. Za formalnu definiciju, pogledati knjigu profesora Vitasa.

Definicija 10

Jezik je $LR(1)$ jezik ako postoji $LR(1)$ gramatika koja ga generiše.

$LR(1)$ gramatike

Odlučivost

- Kao i kod LL -jezika, i ovde je pitanje da li je data gramatika $LR(1)$ odlučivo, ali pitanje da li je jezik $LR(1)$ nije!
 - dakle, ni ovde ne postoji opšti postupak za transformaciju proizvoljne gramatike u ekvivalentnu $LR(1)$ gramatiku ili utvrđivanja da takva gramatika ne postoji
 - sa druge strane, postoji opšti postupak za ispitivanje da li je data gramatika $LR(1)$ i formiranje odgovarajućeg determinističkog proširenog automata, ako jeste
- Kao i kod LL -jezika, i ovde je moguće razmatrati i $LR(k)$ gramatike:
 - ipak, ovde se ispostavi da važi
$$LR(1) = LR(2) = \dots = LR(k) = \dots = DKS$$
 - dakle, analiza naviše omogućava da se uz pomoć samo jednog preduvidnog simbola deterministički parsiraju svi deterministički kontekstno slobodni jezici

SLR(1) gramatike

SLR(1) gramatike

- U opštem slučaju, postupak formiranja determinističkog parsera za proizvoljne $LR(1)$ gramatike je prilično složen
 - Postoji objašnjen u knjizi profesora Vitasu, kao i u Aho-Ulman-u
- Često se razmatraju neke podklase $LR(1)$ gramatika za koje je ovaj postupak značajno jednostavniji, a koje su i dalje dovoljno izražajne da pokriju većinu praktičnih primena
- Jedna takva podklasa je i $SLR(1)$ klasa gramatika (*simple* $LR(1)$)
 - mi ćemo se, u nastavku, prevashodno baviti ovom klasom gramatika

Konačni autotomat za $SLR(1)$ analizu

Definicija 11

Ajtem je pravilo oblika $A \rightarrow \alpha.\beta$, gde je $A \rightarrow \alpha\beta$ pravilo gramatike G . Simbol $.$ je oznaka progrusa.

Napomene

- Ajtem nam, intuitivno, govori dokle smo stigli u prepoznavanju desne strane nekog pravila:
 - deo pre tačke je ono što se nalazi na vrhu steka
 - deo iza tačke je ono što je još potrebno da se prebací na stek, da bismo mogli da izvršimo redukciju po tom pravilu
- Problem je što mi ne znamo unapred po kom pravilu ćemo izvršiti redukciju
- Zbog toga je potrebno paralelno održavati informaciju o napretku u svim pravilima za koja u datom trenutku to ima smisla
 - ovu informaciju o napretku možemo razumeti kao **trenutno stanje**
 - kako pravila ima konačno mnogo, imaćemo konačno mnogo različitih stanja
 - dodavanjem sledećeg simbola na stek, mi prelazimo u sledeće stanje
 - otuda, za opisivanje progrusa u prepoznavanju desnih strana pravila tokom parsiranja možemo koristiti **konačni automat** nad $(\Sigma \cup N)^*$
 - ovaj konačni automat će nam omogućiti da formiramo **tablice prelaska** za parsiranje naviše

Konačni automat za $SLR(1)$ analizu

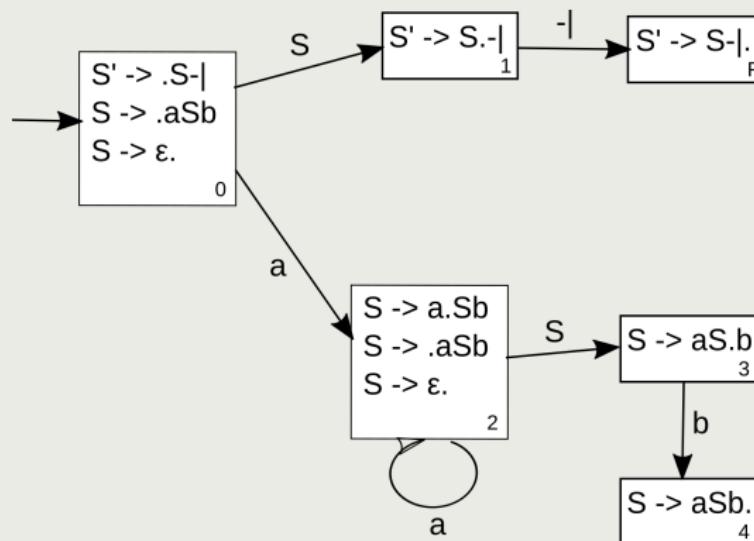
Konačni automat za $SLR(1)$ analizu

- Stanja ovog konačnog automata biće skupovi ajtema
 - ajtem $S' \rightarrow .S \dashv$ naziva se početni ajtem
- princip zatvorenja: ako za neko $B \in N$ ajtem $A \rightarrow \alpha.B\gamma$ pripada nekom stanju s , tada za svako pravilo $B \rightarrow \beta$ gramatike, ajtem $B \rightarrow .\beta$ takođe pripada s
- početno stanje automata je stanje koje nastaje primenom principa zatvorenja na početni ajtem
- princip prelaza: ako za neko $x \in (\Sigma \cup N \cup \{\dashv\})$ ajtem $A \rightarrow \alpha.x\beta$ pripada nekom stanju s , tada ajtem $A \rightarrow \alpha x.\beta$ pripada stanju u koje se prelazi iz s po simbolu x
- završno stanje je stanje koje sadrži ajtem $S' \rightarrow S \dashv$.

Primer

Primer

Ponovo razmatramo gramatiku $S \rightarrow aSb \mid \epsilon$. Konačni automat za SLR(1) analizu izgleda ovako:



Tablice prelaska

Tablica prelaska

- Tablica prelaska za $SLR(1)$ analizu (poznata i kao ACTION-GOTO tablica) se sastoji iz:
 - vrsta koja odgovaraju stanjima konačnog automata za $SLR(1)$ analizu
 - kolona koje odgovaraju terminalima iz $\Sigma \cup \{\vdash\}$ (ACTION deo tablice)
 - kolona koje odgovaraju neterminalima iz N (GOTO deo tablice)
- Označimo sa $T[s, X]$ vrednost u vrsti koja odgovara stanju s i koloni koja odgovara simbolu X :
 - ako u automatu postoji prelaz iz stanja s u stanje t po terminalu $a \in \Sigma \cup \{\vdash\}$, tada u polje $T[s, a]$ upisujemo akciju prebacivanja, u oznaci S_t
 - ako u automatu postoji prelaz iz stanja s u stanje t po neterminalu $A \in N$, tada u polje $T[s, A]$ unosimo akciju potiskivanja, u oznaci P_t
 - ako u stanju s postoji ajtem $A \rightarrow \alpha..$, a simbol $a \in \Sigma \cup \{\vdash\}$ pripada skupu $Sledi(A)$, tada u polje $T[s, a]$ upisujemo akciju redukcije po pravilu $A \rightarrow \alpha$, u oznaci R_i , gde je i redni broj pravila $A \rightarrow \alpha$ u gramatici G

SLR(1) gramatike

Konflikti

Ako, u skladu sa prethodnim algoritmom, u isto polje tabele treba da upišemo:

- i akciju prebacivanja i akciju redukcije, tada imamo konflikt prebacivanje-redukcija (engl. *shift-reduce conflict*)
- dve različite akcije redukcije, tada imamo konflikt redukcija-redukcija (engl. *reduce-reduce conflict*)

Definicija 12

Gramatika G je *SLR(1) gramatika* ako u tablici prelaska formiranoj na prethodno opisan način nema konflikta. Jezik je *SLR(1) jezik* ako postoji *SLR(1) gramatika* koja ga generiše.

Napomene

- Može se pokazati da ovako definisan pojam *SLR(1) gramatika* ne uključuje sve *LR(1) gramatike*
- Takođe, postoje *LR(1) jezici* koji nisu *SLR(1) jezici*
- U praksi, *SLR(1) gramatike* su dovoljne za opis većine programskih i drugih računarskih jezika

Primer

Primer

Vratimo se na prethodni primer i SLR(1) automat za gramatiku $S \rightarrow aSb \mid \epsilon$. Numerišimo pravila proširene gramatike na sledeći način:

$$0: S' \rightarrow S \dashv$$

$$1: S \rightarrow aSb$$

$$2: S \rightarrow \epsilon$$

Tablica prelaska koja odgovara dobijenom automatu je:

| | a | b | \dashv | S |
|---|-------|-------|----------|-------|
| 0 | S_2 | R_2 | R_2 | P_1 |
| 1 | - | - | S_F | - |
| 2 | S_2 | R_2 | R_2 | P_3 |
| 3 | - | S_4 | - | - |
| 4 | - | R_1 | R_1 | - |

Kako nema konflikta (svako polje sadrži najviše po jednu akciju), sledi da je gramatika SLR(1).

Algoritam *SLR(1)* parsiranja

Algoritam

Formiramo proširenji potisni automat na sledeći način:

- Azbuka steka Γ potisnog automata se sastoji iz uređenih parova (x, s) (koje ćemo označavati kraće sa x_s), gde je $x \in \Sigma \cup \{\vdash\} \cup N \cup \{\perp\}$, a s je stanje *SLR(1)* konačnog automata. Pritom, početni simbol steka je par $(\perp, 0)$, gde je \perp marker dna steka, a 0 je početno stanje *SLR(1)* konačnog automata. Na početku rada se na ulazu nalazi niska $w \dashv$, gde je $w \in \Sigma^*$ reč koju parsiramo
- Neka je u nekom trenutku na vrhu steka x_s , a na ulazu $a \in \Sigma \cup \{\vdash\}$. Tada:
 - ako je $T[s, a] = S_t$, tada vršimo **prebacivanje**, tj. sa ulaza skidamo a i na vrh steka postavljamo a_t
 - ako je $T[s, a] = R_i$, tada vršimo akciju redukcije po pravilu $i : A \longrightarrow \gamma$:
 - skidamo sa steka $|\gamma|$ simbola, nakon čega na vrhu steka ostaje neko y_t
 - neka je $T[t, A] = P_{t'}$; na stek potiskujemo $A_{t'}$
 - aku u bilo kom trenutku imamo nedefinisanu vrednost u tablici, prijavljujemo **sintaksnu grešku**
 - kada se na vrhu steka nađe \dashv_F , tada prihvatomо pročitanu reč
 - formalno, u tom trenutku stek nije prazan, već sadrži $\perp_0 S_1 \dashv_F$, ali se to lako može rešiti dodavanjem specijalnog prelaza koji skida ova tri simbola sa steka, ili proglašavanjem ove konfiguracije za završnu; u praksi, to nije suštinski bitno
 - redosled redukcija u obrnutom poretku nam određuje pravila koja treba primeniti u najdešnjem izvođenju niske w

Primer

Primer

Vratimo se opet na gramatiku $S \rightarrow aSb \mid \varepsilon$. Posmatrajmo nisku $aabb$.

Prepoznavanje ove niske se može prikazati sledećom tablicom:

| Stek | Ulaz | Primjenjena akcija |
|---------------------------|---------------|--------------------|
| \perp_0 | $aabb \dashv$ | - |
| $\perp_0 a_2$ | $abb \dashv$ | S_2 |
| $\perp_0 a_2 a_2$ | $bb \dashv$ | S_2 |
| $\perp_0 a_2 a_2 S_3$ | $bb \dashv$ | R_2 |
| $\perp_0 a_2 a_2 S_3 b_4$ | $b \dashv$ | S_4 |
| $\perp_0 a_2 S_3$ | $b \dashv$ | R_1 |
| $\perp_0 a_2 S_3 b_4$ | \dashv | S_4 |
| $\perp_0 S_1$ | \dashv | R_1 |
| $\perp_0 S_1 \dashv_F$ | ε | S_F |

Primer

Primer

Neka je data gramatika izraza:

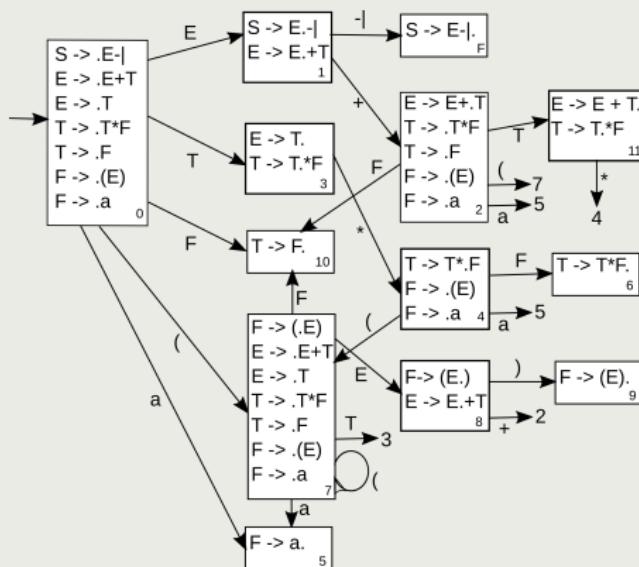
$$\begin{array}{l} 1 : E \longrightarrow E + T \\ 2 : \quad\quad\quad | \quad T \\ 3 : T \longrightarrow T * F \\ 4 : \quad\quad\quad | \quad F \\ 5 : F \longrightarrow (E) \\ 6 : \quad\quad\quad | \quad a \end{array}$$

i reč $a + a * a$. Proširimo ovu gramatiku pravilom $S \longrightarrow E \dashv$, gde je S novi početni simbol. Ispitajmo da li je ova gramatika SLR(1).

Primer

Primer

(nastavak) Automat za SLR(1) analizu izgleda ovako:



Primer

Primer

(nastavak) Setimo se skupova sledi za ovu gramatiku (sa slajda 31): $Sledi(E) = \{+, \), \neg\}$,
 $Sledi(T) = \{+, \), \neg, *\}$ i $Sledi(F) = \{+, \), \neg, *\}$

Imajući to u vidu, SLR(1) ACTION-GOTO tablica izgleda ovako:

| | \neg | a | $+$ | $*$ | $($ | $)$ | E | T | F |
|----|--------|-------|-------|-------|-------|-------|-------|----------|----------|
| 0 | - | S_5 | - | - | S_7 | - | P_1 | P_3 | P_{10} |
| 1 | S_F | - | S_2 | - | - | - | - | - | - |
| 2 | - | S_5 | - | - | S_7 | - | - | P_{11} | P_{10} |
| 3 | R_2 | - | R_2 | S_4 | - | R_2 | - | - | - |
| 4 | - | S_5 | - | - | S_7 | - | - | - | P_6 |
| 5 | R_6 | - | R_6 | R_6 | - | R_6 | - | - | - |
| 6 | R_3 | - | R_3 | R_3 | - | R_3 | - | - | - |
| 7 | - | S_5 | - | - | S_7 | - | P_8 | P_3 | P_{10} |
| 8 | - | - | S_2 | - | - | S_9 | - | - | - |
| 9 | R_5 | - | R_5 | R_5 | - | R_5 | - | - | - |
| 10 | R_4 | - | R_4 | R_4 | - | R_4 | - | - | - |
| 11 | R_1 | - | R_1 | S_4 | - | R_1 | - | - | - |

Kako nema konflikata u tablici, gramatika je SLR(1).

Primer

Primer

(nastavak) Simulacija prepoznavanja niske $a + a * a$ pomoću opisanog parsera data je sledećom tabelom:

| Stek | Ulaz | Primjenjena akcija |
|----------------------------------|--------------------|--------------------|
| \perp_0 | $a + a * a \dashv$ | - |
| $\perp_0 a_5$ | $+ a * a \dashv$ | S_5 |
| $\perp_0 F_{10}$ | $+ a * a \dashv$ | R_6 |
| $\perp_0 T_3$ | $+ a * a \dashv$ | R_4 |
| $\perp_0 E_1$ | $+ a * a \dashv$ | R_2 |
| $\perp_0 E_1 +_2$ | $a * a \dashv$ | S_2 |
| $\perp_0 E_1 +_2 a_5$ | $* a \dashv$ | S_5 |
| $\perp_0 E_1 +_2 F_{10}$ | $* a \dashv$ | R_6 |
| $\perp_0 E_1 +_2 T_{11}$ | $* a \dashv$ | R_4 |
| $\perp_0 E_1 +_2 T_{11} *_4$ | $a \dashv$ | S_4 |
| $\perp_0 E_1 +_2 T_{11} *_4 a_5$ | \dashv | S_5 |
| $\perp_0 E_1 +_2 T_{11} *_4 F_6$ | \dashv | R_6 |
| $\perp_0 E_1 +_2 T_{11}$ | \dashv | R_3 |
| $\perp_0 E_1$ | \dashv | R_1 |
| $\perp_0 E_1 \dashv_F$ | ε | S_F |

Primer

Primer

(nastavak) Pogledajmo šta bi se desilo kada bismo na ulazu prethodnog automata imali nisku $a * (a + (a + a))$ (primetimo da nedostaje jedna zagrada na kraju):

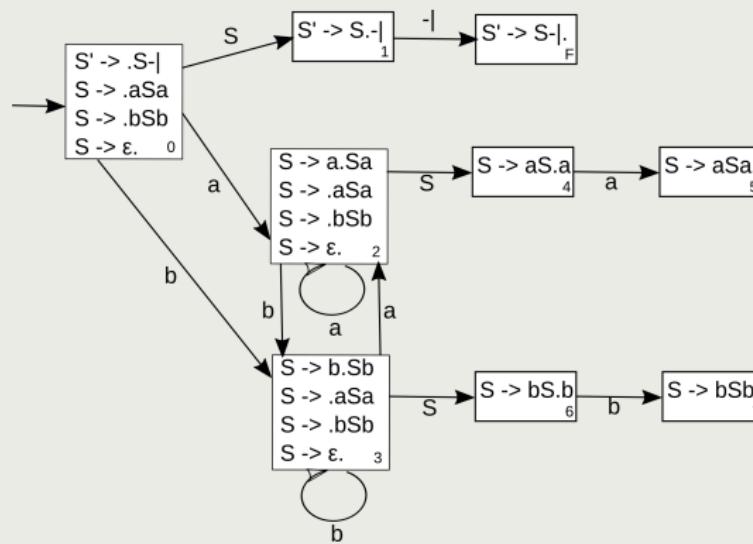
| Stek | Ulaz | Primjenjena akcija |
|--|----------------------------|--------------------|
| \perp_0 | $a * (a + (a + a)) \dashv$ | - |
| $\perp_0 a s$ | $*(a + (a + a)) \dashv$ | S_5 |
| $\perp_0 F_{10}$ | $*(a + (a + a)) \dashv$ | R_6 |
| $\perp_0 T_3$ | $*(a + (a + a)) \dashv$ | R_4 |
| $\perp_0 T_3 * 4$ | $(a + (a + a)) \dashv$ | S_4 |
| $\perp_0 T_3 * 4 (7$ | $a + (a + a)) \dashv$ | S_7 |
| $\perp_0 T_3 * 4 (7 a_5$ | $+(a + a)) \dashv$ | S_5 |
| $\perp_0 T_3 * 4 (7 F_{10}$ | $+(a + a)) \dashv$ | R_6 |
| $\perp_0 T_3 * 4 (7 T_3$ | $+(a + a)) \dashv$ | R_4 |
| $\perp_0 T_3 * 4 (7 E_8$ | $+(a + a)) \dashv$ | R_2 |
| $\perp_0 T_3 * 4 (7 E_8 + 2$ | $(a + a)) \dashv$ | S_2 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7$ | $a + a)) \dashv$ | S_7 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 a_5$ | $+a)) \dashv$ | S_5 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 F_{10}$ | $+a)) \dashv$ | R_6 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 T_3$ | $+a)) \dashv$ | R_4 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 E_8$ | $+a)) \dashv$ | R_2 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 E_8 + 2$ | $a) \dashv$ | S_2 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 E_8 + 2 a_5$ | $) \dashv$ | S_5 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 E_8 + 2 F_{10}$ | $) \dashv$ | R_6 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 E_8 + 2 T_{11}$ | $) \dashv$ | R_4 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 E_8$ | $) \dashv$ | R_1 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 (7 E_8)_9$ | \dashv | S_9 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 F_{10}$ | \dashv | R_6 |
| $\perp_0 T_3 * 4 (7 E_8 + 2 T_{11}$ | \dashv | R_4 |
| $\perp_0 T_3 * 4 (7 E_8$ | \dashv | R_1 |

Sada za stanje 8 i simbol \dashv ne postoji prelaz u tablici, tako da na ovom mestu sintaksni analizator prijavljuje grešku.

Primer

Primer

Posmatrajmo gramatiku $S \rightarrow aSa \mid bSb \mid \epsilon$. Konačni automat za SLR(1) analizu dat je na slici:



Primer

Primer

(nastavak) S obzirom da je $\text{Prvi}(S) = \{a, b\}$, a $\text{Sledi}(S) = \{\dashv, a, b\}$, tablica prelaska za $SLR(1)$ analizu (ACTION-GOTO tablica) izgleda ovako:

| | a | b | \dashv | S |
|---|-----------|-----------|----------|-------|
| 0 | S_2/R_3 | S_3/R_3 | R_3 | P_1 |
| 1 | - | - | S_F | - |
| 2 | S_2/R_3 | S_3/R_3 | R_3 | P_4 |
| 3 | S_2/R_3 | S_3/R_3 | R_3 | P_6 |
| 4 | S_5 | - | - | - |
| 5 | R_1 | R_1 | R_1 | - |
| 6 | - | S_7 | - | - |
| 7 | R_2 | R_2 | R_2 | - |

Dakle, vidimo da postoje konflikti prebacivanje-redukcija u poljima $T[0, a]$, $T[0, b]$, $T[2, a]$, $T[2, b]$, $T[3, a]$ i $T[3, b]$. Otuda, gramatika nije $SLR(1)$.

Višeznačne gramatike

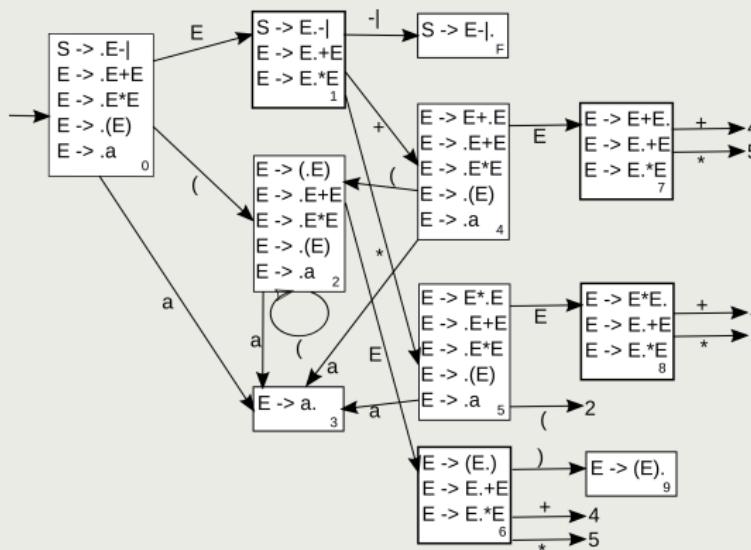
Više značne gramatike

- Višeznačne gramatike nisu $LR(1)$, s obzirom da izvođenje nadesno ne mora biti jedinstveno
- Otuda bi primena opisanog postupka za formiranje $SLR(1)$ parsera definitivno vodila ka pojavi konflikata u tablici prelaska
- Ipak, to ne znači da se alati za analizu naviše ne mogu koristiti za parsiranje višeznačnih gramatika:
 - višeznačnosti se tipično javljaju kao posledica nerazrešenih prioriteta i asocijativnosti
 - većina alata ostavlja mogućnost da se prioriteti i asocijativnosti navedu eksplisitno
 - ove informacije se mogu koristiti pri pojavi konflikta, kako bi se donela odluka o sledećem koraku

Primer

Primer

Posmatrajmo višeznačnu gramatiku izraza $E \rightarrow E + E \mid E * E \mid (E) \mid a$. Automat za SLR(1) analizu dat je na sledećoj slici:



Primer

Primer

(nastavak) S obzirom da je $\text{Prvi}(E) = \{(, a\}$ i $\text{Sledi}(E) = \{\dashv, +, *, \}\},$ imamo sledeću tablicu prelaska:

| | a | + | * | (|) | \dashv | E |
|---|-------|-----------|-----------|-------|-------|----------|-------|
| 0 | S_3 | - | - | S_2 | - | - | P_1 |
| 1 | - | S_4 | S_5 | - | - | S_F | - |
| 2 | S_3 | - | - | S_2 | - | - | P_6 |
| 3 | - | R_4 | R_4 | - | R_4 | R_4 | - |
| 4 | S_3 | - | - | S_2 | - | - | P_7 |
| 5 | S_3 | - | - | S_2 | - | - | P_8 |
| 6 | - | S_4 | S_5 | - | S_9 | - | - |
| 7 | - | S_4/R_1 | S_5/R_1 | - | R_1 | R_1 | - |
| 8 | - | S_4/R_2 | S_5/R_2 | - | R_2 | R_2 | - |
| 9 | - | R_3 | R_3 | - | R_3 | R_3 | - |

Tablica, očekivano, sadrži konflikte.

Primer

Primer

(nastavak) Konflikti koji postoje u tablici su posledica nedefinisanih prioriteta i asocijativnosti i mogu se razrešiti eksplicitnim navođenjem:

- konflikt u polju $T[7, +]$ je posledica nedefinisane asocijativnosti operatora $+$:
 - ako želimo levu asocijativnost, daćemo prednost redukciji
 - ako želimo desnu asocijativnost, daćemo prednost prebacivanju
- konflikt u polju $T[7, *]$ je posledica nedefinisanih prioriteta operatora:
 - ako želimo da operator $*$ ima viši prioritet, daćemo prednost prebacivanju
 - ako želimo da operator $+$ ima viši prioritet, daćemo prednost redukciji
- konflikt u polju $T[8, +]$ je posledica nedefinisanih prioriteta operatora:
 - ako želimo da operator $*$ ima viši prioritet, daćemo prednost redukciji
 - ako želimo da operator $+$ ima viši prioritet, daćemo prednost prebacivanju
- konflikt u polju $T[8, *]$ je posledica nedefinisane asocijativnosti operatora $*$:
 - ako želimo levu asocijativnost, daćemo prednost redukciji
 - ako želimo desnu asocijativnost, daćemo prednost prebacivanju

Asocijativnost i prioritet u YACC-u

Zadavanje prioriteta i asocijativnosti u YACC-u

Alat **YACC** (i njegova slobodna varijanta **Bison**) podržava eksplisitno definisanje prioriteta i asocijativnosti operatora:

- Direktiva `%left '+'` označava levu asocijativnost operatora +
- Direktiva `%right '+'` označava desnu asocijativnost operatora +

Prioriteti se definišu redosledom navođenja ovih direktiva:

- operatori uvedeni ranije imaju niži prioritet
- operatori uvedeni kasnije imaju viši prioritet

Direktive `%left` i `%right` se navode pre gramatike, u odeljku u kome se definišu tokeni (koriste se umesto direktive `%token` kojom se definišu tokeni koji nemaju asocijativnost).

Primer

*U prethodnom primeru, ako želimo da oba operatora imaju levu asocijativnost, a da operator * ima viši prioritet, u YACC-u ćemo navesti:*

```
%left '+'  
%left '*'  
...
```

LR(1) i SLR(1) (ne ide za ispit)

LR(1) i SLR(1), ponovo

- Prilikom formiranja konačnog automata za *SLR(1)* analizu uopšte nismo uzimali u obzir preduvidne simbole, već su stanja formirana isključivo na osnovu pravila gramatike:
 - zbog toga se konačni automat za *SLR(1)* analizu u literaturi često naziva i *LR(0)*-automat, a ajtemi ovog automata i *LR(0)*-ajtemi
- Konflikti između prebacivanja i redukcije su rešavani pomoću skupova *Sledi*:
 - ispostavi se da je ovo prilično gruba aproksimacija onoga što se zaista dešava prilikom parsiranja, jer je skup preduvidnih simbola za koje je potrebno vršiti redukciju po pravilu $A \rightarrow \gamma$ obično neki podskup skupa *Sledi*(A)
 - na žalost, ovaj podskup nije moguće odrediti nakon što se formira automaton, već se informacije o ovim skupovima moraju održavati tokom formiranja automata

LR(1) i SLR(1) (ne ide za ispit)

LR(1) ajtemi

- **LR(1) ajtem:** ajtem oblika $A \longrightarrow \alpha.\beta(F)$ čije je značenje „prepoznali smo α , ostaje još da prepoznamo β , nakon čega možemo vršiti redukciju po ovom pravilu ukoliko je na ulazu neko $a \in F \subseteq \Sigma^*$ “.
- **Skup preduvida** F se formira na osnovu dopunjenoog **principa zatvorenja**: ako u stanju s imamo ajtem $A \longrightarrow \alpha.B\beta(F)$, gde je $B \in N$, tada u stanje s treba dodati i ajtem $B \longrightarrow .\gamma(Prvi(\beta))$ (odnosno $B \longrightarrow .\gamma(Prvi(\beta) \cup F)$ ako je β anulirajuće), za svako B -pravilo gramatike
- Početni ajtem je sada $S' \longrightarrow .S \dashv (\{\})$
- Dva LR(1) ajtema su različita ako su im skupovi preduvida različiti, čak i ako su inače isti (kao LR(0) ajtemi)
 - zbog toga će skup stanja LR(1) automata biti značajno veći nego kod SLR(1) automata
 - ipak, finija analiza mogućih preduvida omogućava razrešavanje nekih konflikata koji kod SLR(1) analize ostaju nerazrešeni
- Detalje pogledati u knjizi profesora Vitasu

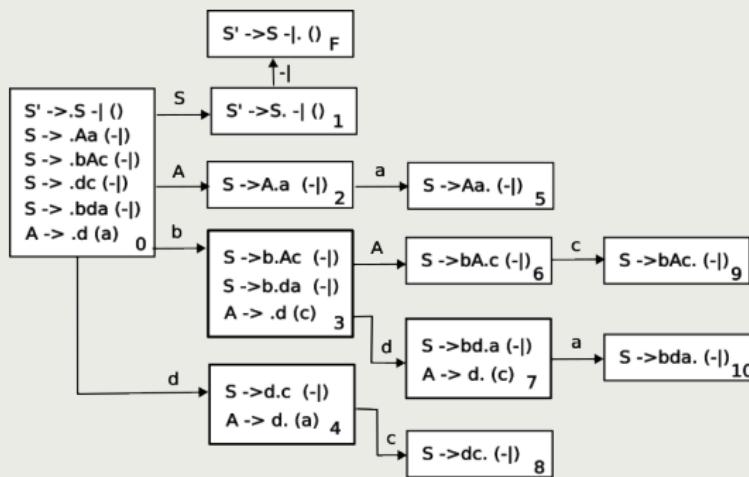
VAŽNA NAPOMENA

U slučaju kanonskog LR(1) parsera, jedino je postupak formiranja tablica prelaska složeniji (zbog većeg broja stanja LR(1) konačnog automata) – jednom kada ove tablice odredimo, algoritam parsiranja je [potpuno isti](#) kao i kod SLR(1) analize (algoritam na slajdu 57).

LR(1) i SLR(1) (ne ide za ispit)

Primer

Posmatrajmo gramatiku $S \rightarrow Aa \mid bAc \mid dc \mid bda$, $A \rightarrow d$. Automat za LR(1) analizu dat je na sledećoj slici:



Iza svakog ajtema, u zagradi stoje preduviđi.

LR(1) i SLR(1) (ne ide za ispit)

Primer

(nastavak) Tablicu prelaska formiramo na sličan način kao kod SLR(1) analize, s tim što se redukcija po završnom ajtemu vrši samo za preduvide pridružene tom ajtemu, a ne za ceo skup Sledi.

| | <i>a</i> | <i>b</i> | <i>c</i> | <i>d</i> | \vdash | <i>S</i> | <i>A</i> |
|----|----------|----------|----------|----------|----------|----------|----------|
| 0 | - | S_3 | - | S_4 | - | P_1 | P_2 |
| 1 | - | - | - | - | S_F | - | - |
| 2 | S_5 | - | - | - | - | - | - |
| 3 | - | - | - | S_7 | - | - | P_6 |
| 4 | R_5 | - | S_8 | - | - | - | - |
| 5 | - | - | - | - | R_1 | - | - |
| 6 | - | - | S_9 | - | - | - | - |
| 7 | S_{10} | - | R_5 | - | - | - | - |
| 8 | - | - | - | - | R_3 | - | - |
| 9 | - | - | - | - | R_2 | - | - |
| 10 | - | - | - | - | R_4 | - | - |

Rezultujuća tablica ne sadrži konflikte, pa je gramatika LR(1). Primetimo da bi u slučaju SLR(1) tablica, shift-reduce konflikt postojao u poljima $T[4, c]$ i $T[7, a]$, s obzirom da je $Sledi(A) = \{a, c\}$. Dakle, gramatika nije SLR(1).