ჯემალ ანთიძე

ფორმალური ენები და გრამატიკები

სახელმძღვანელო კომპიუტერულ მეცნიერებათა

მიმართულების სტუდენტებისათვის

თბილისი 2017

ი.ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ფორმალური ენები და გრამატიკები

სახელმძღვანელო კომპიუტერულ მეცნიერებათა მიმართულების სტუდენტებისათვის

ჯემალ ანთიძე

6/15/2017

Comment [j1]:

Comment [j2]:

Comment [j3]:

Comment [j4]:

Comment [j5]:

Comment [j6]:

Comment [j7]:

Comment [j8]:

Comment [j9]:

Comment [j10]:

სახელმძღვანელოში გადმოცემულია ფორმალურ გრამატიკათა თანამედროვე თეორია ხომსკის კლასიფიკაციით. აღწერილია კონტექსტისაგან თავისუფალი გრამატიკის ხომსკისა და გრეიბახის ნორმალური ფორმები, მოცემულია თითოეული კლასის გრამატიკისათვის გარჩევის ალგორითმი. მოყვანილია მონტეგიუს გრამატიკის PTQ ფრაგმენტი და მისი გამოყენება ბუნებრივი ენების კომპიუტერული დამუშავებისათვის. აგრეთვე, აღწერილია სასრული ავტომატები და ავტომატები მაღაზიური მეხსიერებით და მათი გამოყენება გარჩევის ალგორითმების კომპიუტერული რეალიზაციისათვის. სახელმძღვანელო სასარგებლო იქნება არა მარტო კომპიუტერული მეცნიერების მიმართულების სტუდენტებისათვის არამედ აგრეთვე მათთვისაც, ვინც დაიტერესებულია რთული სისტემების კომპიუტერული რეალიზაციით. როგორიცაა, დაპროგრამების ენების კომპილატორების შექმნა, ბუნებრივი ენების კომპიუტერული მოდელირება და სხვა ხელოვნური ინტელექტის პრობლემების გადაწყვეტა. ეს, ასეთი სახელმძღვანელოს ქართულ ენაზე გამოცემის პირველი ცდაა და არაა დაზღვეული ნაკლოვანებებისაგან. ავტორი მადლიერებით მიიღებს საქმიან კრიტიკულ შენიშვნებსა თუ სურვილებს.

სარჩევი

1 თავი

ფორმალური ენისა დაგრამატიკის განსაზღვრა, სასრული ავტომატები

1.1 ფორმალური ენის განსაზღვრა

განვიხილოთ რაიმე სიმბოლოების სასრული სიმრავლე Σ, რომელსაც ვუწოდოთ ალფაბეტი. ალფაბეტი გამოიყენება ნებისმიერ ბუნებრივ თუ ფორმალურ ენაში. სიმბოლოს ქვეშ იგულისხმება ნებისმიერი ნიშანი, რომელიც შეიძლება შეგვხვდეს ტექსტში. სიმბოლოთა ნებისმიერ სასრულ მიმდევრობას ვუწოდოთ ჯაჭვი (ზოგჯერ მას უწოდებენ სიტყვას ან სტრიქონს). ისეთ ჯაჭვს, რომელიც არ შეიცავს არცერთ სიმბოლოს ვუწოდოთ ცარიელი ჯაჭვი და ავღნიშნოთ e სიმბოლოთი. ჯაჭვების ნებისმიერ სასრულ მიმდევრობას ვუწოდოთ ტექსტი. იგულისხმება, რომ ჯაჭვები ტექსტში ერთიმეორისაგან გამოყოფილია სპეციალური ხარვეზის ნიშნით. ხარვეზის ნიშნად ფორმალურ ენებში გამოიყენება სიმბოლო # , ხოლო ტექსტში მის აღსანიშნავად გამოიყენება ცარიელი ადგილი. ტექსტში გამოყენებულ ცარიელ ადგილს ჩვენ ჩავთვლით სიმბოლოდ. ხარვეზის ნიშანი არ შეი'ლება შედიოდეს ჯაჭვში, რადგანაც იგი გამოიყენება ჯაჭვების განმაცალკევებლად, მაგრამ იგი ყოველთვის შედის ალფაბეტში. ცალკეულ სიმბოლოებს ზოგადად a, b, c, d ასოებით აღვნიშნავთ, ხოლო ჯაჭვებს t, u, v, w, x, y, და z ასოებით აღვნიშნავთ. c^i-თი აღვნიშნავთ ჯაჭვს, რომელიც i რაოდენობა c სიმბოლოსაგან შედგება. თუ გვაქვს ორი x და y ჯაჭვი, მაშინ xy ჯაჭვს x და y ჯაჭვების კონკატენაცია ეწოდება და y-ის x-ის მარჯვნივ მიწერით მიიღება. ჩვენ არ გამოვიყენებთ სპეციალურ სიმბოლოს კონკატენაციის ოპერაციის აღსანიშნავად. შემოტანილი შეთანხმებებიდან გამომდინარეობს, რომ $a^0=e$ და $a^2=aa$. რეკურსიულად, რაიმე ჯაჭვი Σ ალფაბეტში განისაზღვრება შემდეგნაირად:

1. e არის ჯაჭვი Σ ალფაბეტში;

2. თუ x არის ჯაჭვი Σ ალფაბეტში და $a \in \Sigma$, მაშინ xa არის ჯაჭვი Σ -ში;

3. რაიმე z ჯაჭვი არის ჯაჭვი Σ-ში მხოლოდ და მხოლოდ მაშინ, თუ იგი 1 და 2 წესებით მიიღება.

იგულისხმება, რომ ex=xe=x. თუ $x=c_1c_2...c_n$, მაშინ $c_nc_{n-1}...c_1$ ჯაჭვს ეწოდება x-ის შებრუნებული ჯაჭვი და x^R-ით აღვნიშნავთ. თუ გვაქვს yxz ჯაჭვი, მაშინ y-ს ეწოდება yxz-ის პრეფიქსი, ხოლო z-ს ეწოდება yxz-ის სუფიქსი, ხოლო x-ს ინფიქსი ეწოდება. x ჯაჭვის სიგრ'ე |x| -ით აღვნიშნოთ და განვმარტოთ შემდეგნაირად: თუ x=e, მაშინ |x| = 0; თუ $x=a_1a_2...a_n$ მაშინ |x| =n. მაგალითად, |aabcd| =5. ჯაჭვების L რაიმე სიმრავლეს Σ ალფაბეტში ვუწოდოთ L ენა Σ ალფაბეტში. ეს განმარტება მოიცავს როგორც ფორმალურ ენებს ასევე ბუნებრივ ენებსაც. Σ * -ით აღვნიშნოთ ენა,

რომელიც მოიცავს ყველა ჯაჭვს Σ ალფავიტიდან e ჯაჭვის ჩათვლით, ხოლო Σ^+ -ით აღვნიშნოთ სიმრავლე $\Sigma^* - \{e\}$, ე.ი. Σ^* სიმრავლე ცარიელი ჯაჭვის გარდა. Σ^* -ს უწოდებენ უნივერსალურ ენას Σ ალფაბეტში, რადგან ნებისმიერი L ენა Σ ალფაბეტში არის მისი ქვესიმრავლე. დაპროგრამების FORTRAN ენა არის ენა ჩვენი განმარტებით FORTRAN-ის ალფაბეტში, თუ ჯაჭვებად ავიღებთ სინტაქსურად სწორ პროგრამებს. ასევე ქართული ენა არის ენა ჩვენი განმარტებით ქართულ ალფაბეტში, თუ სწორ ქართულ წინადადებებს ავიღებთ ჯაჭვებად.

1.2 ფორმალური გრამატიკები

როგორც წინა პარაგრაფიდან უკვე ვიცით ნებისმიერი L ფორმალური ენა შეიძლება ჩაიწეროს ასე $L \subseteq \Sigma^*$, სადაც Σ რაიმე ალფაბეტია. ასეთი ენა ყოველთვის უსასრულოა, თუ ჩვენ არ შევზღუდავთ ჯაჭვის სიგრძეს. რადგან ჩვენ შემდგომში დაგვაინტერესებს ისეთი ენები, რომელთა ჯაჭვებს აქვთ სასრული სიგრძე, ამიტომ ჩვენ ყოველთვის ვიგულისხმებთ, რომ $|x| \leq n$, სადაც n არაუარყოფითი მთელი რიცხვია და $x \in L$. რაიმე L ენის მოცემა შეიძლება მისი ჯაჭვების ჩამოთვლით, თუ ენა სასრულია. მაგალითად, ასეთ შემთხვევაშიც კი მოუხერხებელია ენის მოცემა მისი ჯაჭვების ჩამოწერით. ამიტომ, ბუნებრივია მოიძებნოს ისეთი ხერხი, რომელიც სრულად აღწერდა ენას და მარტივი იქნებოდა იმის შესამოწმებლად, ეკუთვნის თუ არა რაიმე x ჯაჭვი ამ ენას. ზოგჯერ, შესაძლებელია L ენა ჩაიწეროს რაიმე ფორმულით, მაგალითად $\{ a^n b^n c^n \mid n \geq 0 \}$. მაგრამ ფორმულით ენის წარმოდგენა ხერხდება იშვიათ შემთხვევებში. ამიტომ, უნდა მოიძებნოს სხვა ხერხი. ერთ-ერთ ასეთ ხერხს წარმოადგენს ენის მოცემა წარმომქმნელი გრამატიკის საშუალებით. მეორე გზას წარმოადგენს ენის მოცემა კერძო ალგორითმით, რომელიც ამთავრებს მუშაობას თუ ჯაჭვი ეკუთვნის ენას. ჩვენ შევჩერდებით პირველ ხერხზე, კერძოდ, ხომსკის გრამატიკებზე. ახლა, ჩვენ შევუდგებით ასეთი გრამატიკების აღწერას ჯერ არაფორმალურად, შემდეგ კი ზუსტად განვმარტავთ მათ.

რაიმე L ენის აღსაწერად განვიხილოთ ორი სიმრავლე. ერთი სიმრავლეა ამ ენის ალფაბეტი Σ , რომელსაც ვუწოდებთ ტერმინალურ სიმბოლოთა სიმრავლეს და მეორე სიმრავლეა სიმბოლოთა N სიმრავლე, რომელიც ასევე სასრულია და ვუწოდებთ არატერმინალურ სიმბოლოთა სიმრავლეს. განვიხილოთ წესების P სასრული სიმრავლე, რომელიც წარმოადგენს (α , β) წყვილთა სიმრავლეს, სადაც

$$(\alpha , \beta) \ (N \cup \Sigma)^* N (N \cup \Sigma)^* X (N \cup \Sigma)^*$$

სიმრავლის რაიმე ელემენტია. ე. ი. წესის პირველი ელემენტი აუცილებლად შეიცავს ერთ არატერმინალურ სიმბოლოს მაინც, მაშინ როცა მეორე ელემენტი შეიძლება იყოს ცარიელი ჯაჭვი. წესი შეიძლება ჩაიწეროს ასეც $\alpha \to \beta$. ჩვენ შემდგომში გამოვიყენებთ წესის ჩაწერის მეორე სახეს. N სიმრავლეში არსებობს სპეციალურად გამოყოფილი სიმბოლო, რომელსაც S-ით აღვნიშნავთ და ვუწოდებთ საწყის სიმბოლოს. ყოველთვის ვიწყებთ L ენის რაიმე ჯაჭვის მიღებას P სიმრავლის რაიმე

წესით, როლმელსაც აქვს სახე $S \to \alpha$ და შემდეგ α -ში ($\alpha = \alpha_1 \beta \alpha_2$) მის რაიმე ქვეჯაჭვს β -ს ვცვლით γ -თი, სადაც $\beta \to \gamma$ P –ს რაიმე წესია. მივიღებთ ჯაჭვს $\alpha_1 \gamma \alpha_2$ ასევე ვიქცევით ამ ჯაჭვის მიმართაც, სანამ არ დასრულდება ეს პროცესი. თუ შედეგად მივიღეთ x ჯაჭვი, რომელიც მხოლოდ ტერმინალურ სიმბოლოებს შეიცავს, მაშინ მას ვაცხადებთ L ენის x ჯაჭვად. ყველა ასეთი x ჯაჭვების ერთობლიობა განსაზღვრავს L ენას და მას ეწოდება ზემოთ აღწერილი გრამატიკით განსაზღვრული ენა. ზემოთ აღწერილი გრამატიკა ფორმალურად განისაზღვრება შემდეგნაირად: G=(N, ∑, P, S) ოთხეულს ვუწოდებთ გრამატიკას, სადაც

1. N - არატერმინალური სიმბოლოების სასრული სიმრავლეა;

2. ∑ - ტერმინალურ სიმბოლოთა სასრული სიმრავლეა და $N \cap \sum = \varnothing$ (N -ის გადაკვეთა ∑ -სთან ცარიელი სიმრავლეა);

3. P არის $(N \cup \sum) * N (N \cup \sum) * X (N \cup \sum) *$ სიმრავლის სასრული ქვესიმრავლე;

4. $S \in N$ გამოყოფილი სიმბოლოა, რომელსაც საწყისი სიმბოლო ეწოდება.

G გრამატიკა განსაზღვრავს L(G) ენას, რომელსაც G გრამატიკით განსაზღვრული ენა ეწოდება. შემოვიტანოთ რეკურსიულად სპეციალური სახის ჯაჭვები, რომელთაც G გრამატიკის გამოყვანადი ჯაჭვები ეწოდებათ:

1. S - გამოყვანადი ჯაჭვია;

2. თუ $\alpha \beta \gamma$ - გამოყვანადი ჯაჭვია და $\beta \to \delta$ ელემენტია P- სი, მაშინ $\alpha \delta \gamma$ გამოყვანადი ჯაჭვია, აგრეთვე.

გამოყვანადი ჯაჭვი, რომელიც არ შეიცავს არატერმინალურ სიმბოლოებს ეწოდება G გრამატიკით წარმოქმნილი ტერმინალური ჯაჭვი. ტერმინალურ ჯაჭვთა ერთობლიობა ქმნის G გრამატიკით წარმოქმნილ L ენას და აღინიშნება L (G)-ით. ახლა შემოვიტანოთ $\Rightarrow$ დამოკიდებულება იმისათვის, რომ მარტივად ჩავწეროთ L(G) ენა. თუ $\alpha_1 \beta \alpha_2$ გამოყვანადი ჯაჭვია G გრამატიკაში და $\beta \to \gamma \in P$, მაშინ $\alpha_1 \beta \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2$.

$\Rightarrow$ დამოკიდებულების ტრანზიტული ჩაკეტვა ავღნიშნოთ $\Rightarrow+$-ით, ხოლო ტრანზიტული და რეფლექსური ჩაკეტვა $\Rightarrow*$-თი. ამის შემდეგ, L(G) ენა შეიძლება ჩაიწეროს შემდეგნაირად: $L(G) = \{ w \mid w \in \sum *, S \Rightarrow * w \}$. განვიხილოთ გრამატიკის მაგალითი, რომელიც განსაზღვრავს არითმეტიკულ გამოსახულებათა ენას, რომელიც შეიცავს არითმეტიკულ ოპერაციებს შეკრებას და გამრავლებას, იდენტიფიკატორს a-ს და მრგვალ ფრჩხილებს. ვთქვათ $G_1 = (\{ E , T , F \} , \{ a, + , * , (,) \} , P , E)$, სადაც P შეიცავს წესებს

$E \rightarrow E + T \mid T$
$T \rightarrow T * E \mid F$
$F \rightarrow (E) \mid a$

მაგალითად, ჯაჭვი (a+ a)* a ეკუთვნის $L(G_1)$-ს, რის ჩვენებაც მარტივია შემდეგი გამოყვანით:
$E \Rightarrow T \Rightarrow T * F \Rightarrow F* F \Rightarrow (E) * F \Rightarrow (E + T) * F \Rightarrow (T + T) * F \Rightarrow (F+ T) * F \Rightarrow ($
$a+ T) * F \Rightarrow (a+ F)* F \Rightarrow (a+ a)+F \Rightarrow (a+a)* a.$

შევნიშნოთ, რომ არსებობს სხვა გამოყვანებიც, რომლებიც მოგვცემენ იგივე შედეგს. ისინი მიიღებიან გამოყენებული წესების რიგის შეცვლით, როცა არსებობს რამოდენიმე ალტერნატივა. ეს გრამატიკა საინტერესოა იმით, რომ თუ ჩვენ მას დავუმატებთ ახალ წესებს სხვა არითმეტიკული ოპერაციებისათვის მათი პრიორიტეტების გათვალისწინებით, შეიძლება მივიღოთ გრამატიკა პროგრამირების ფართოდ ცნობილ ენებში არითმეტიკული გამოსახულების განსასაზღვრავად. ამ მაგალითში ჩვენ გამოვიყენეთ წესების შემოკლებული ჩაწერა. საზოგადოდ, თუ ჩვენ
გვაქვს წესები:

$\alpha \rightarrow \beta_1$
$\alpha \rightarrow \beta_2$
.
.
.
$\alpha \rightarrow \beta_n$

შემოკლებით იგი ასე ჩაიწერება:

$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n.$

1.3 გრამატიკათა ხომსკის კლასიფიკაცია

Comment [j11]:

ვთქვათ, მოცემულია გრამატიკა G=(N , Σ , P , S). თუ ჩვენ P-ს წესებს დავადებთ გარკვეულ შეზღუდვებს, მივიღებთ სხვადასხვა ტიპის გრამატიკებს, რომლებიც ფართოდ ცნობილია ფორმალურ გრამატიკათა თეორიაში და შემოტანილია ხომსკის მიერ. გრამატიკას ეწოდება მარჯვნივ წრფივი, თუ მის ყოველ წესს აქვს სახე:

$A \rightarrow XB$ ან $A \rightarrow X$, სადაც $A,B \in N$ და $X \in \Sigma^*$.

გრამატიკას ეწოდება კონტექსტისაგან თავისუფალი ან კონტექსტისადმი არამგრძნობიარე თუ მის ყოველ წესს აქვს სახე: $A \rightarrow \alpha$, სადაც $A \in N$,

$\alpha \in (N \cup \Sigma)^*$. გრამატიკას ეწოდება კონტექსტზე დამოკიდებული ან კონტექსტისადმი მგრძნობიარე, ზოგჯერ არადამოკლებადი თუ მის ყოველ წესს აქვს სახე $\alpha \to \beta$, სადაც $|\alpha| \leq |\beta|$.

თვითეული ტიპის გრამატიკა განსაზღვრავს გარკვეული კლასის ენებს, რომელთაც გარკვეული თვისებები გააჩნიათ. საზოგადოდ შევნიშნოთ, რომ კონკრეტული L ენა განისაზღვრება ცალსახად რაიმე გრამატიკით, ე. ი. თუ L ენა განისაზღვრება G გრამატიკით ეს იმას არ ნიშნავს, რომ არ არსებობს რაიმე G_1 გრამატიკა G გრამატიკისაგან განსხვავებული, რომელიც იგივე L ენას განსაზღვრავს. ან სხვანაირად, თუ $L(G_1)= L(G_2)$, მიუხედავად ამისა, თუ L ენა განისაზღვრება რაიმე ტიპის გრამატიკით, ჩვენ შეგვილია ვილაპარაკოთ L ენის გარკვეულ თვისებებზე. დავიწყოთ უმარტივესი გრამატიკებით - მარჯვნივ წრფივი გრამატიკებით და ვაჩვენოთ, რომ მათ მიერ განსაზღვრული ენები არიან რეგულარული სიმრავლეები და პირიქით, ნებისმიერი რეგულარული სიმრავლე განისაზღვრება რაიმე მარჯვნივ წრფივი გრამატიკით.

1.4 რეგულარული სიმრავლეები

ვთქვათ, მოცემულია Σ სასრული ალფაბეტი. რეგულარული სიმრავლე Σ ალფაბეტში რეკურსიულად განისაზღვრება შემდეგნაირად:

1. ცარიელი სიმრავლე - $\varnothing$ არის რეგულარული სიმრავლე Σ ალფაბეტში;

2. {e} არის რეგულარული სიმრავლე Σ ალფაბეტში(e ცარიელ სიმბოლოს აღნიშნავს);

3. { a} არის რეგულარული სიმრავლე Σ ალფაბეტში, სადაც $a \in \Sigma$;

4. თუ P და Q რეგულარული სიმრავლეებია Σ ალფაბეტში, მაშინ რეგულარული სიმრავლეებია, აგრეთვე, $P \cup Q$ PQ P^*;

5. რეგულარული სიმრავლეებია, მხოლოდ ის სიმრავლეები, რომლებიც მიიღებიან 1-4 წესებით. რეგულარული სიმრავლეები გამოისახებიან რეგულარული გამოსახულებებით. რეგულარული გამოსახულება Σ ალფაბეტში მოიცემა შემდეგი რეკურსიული განსაზღვრებებით:

1. ϕ აღნიშნავს რეგულარულ სიმრავლეს ϕ (ϕ ცარიელი სიმრავლეა).

2. e აღნიშნავს რეგულარულ სიმრავლეს {e}.

Comment [j12]:
Comment [j13]:
Comment [j14]:
Comment [j15]:
Comment [j16]:
Comment [j17]:

3. a აღნიშნავს რეგულარულ სიმრავლეს {a}, სადაც $a \in \Sigma$.

4. თუ p და q რეგულარული გამოსახულებებია, რომლებიც აღნიშნავენ რეგულარულ სიმრავლეებს P-ს და Q-ს შესაბამისად, მაშინ (p+q), (pq) და (p)* აგრეთვე რეგულარული გამოსახულებებია, რომლებიც აღნიშნავენ რეგულარულ სიმრავლეებს PUQ, PQ და P* შესაბამისად.

5. რეგულარული გამოსახულებებია მხოლოდ ის გამოსახულებები, რომლებიც მიიღებიან 1 – 4 წესებით.

რეგულარულ გამოსახულებათათვის α , β და γ სამართლიანია შემდეგი ტოლობები, რომელთა დამტკიცება ადვილია:

$\alpha + \beta = \beta + \alpha$

$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$

$\alpha (\beta + \gamma) = \alpha \beta + \alpha \gamma$

$\alpha e = e\alpha = \alpha$

$\phi * = e$

$\alpha * = \alpha + \alpha *$

$\alpha (\beta \gamma) = (\alpha \beta) \gamma$

$\alpha + \alpha = \alpha$

$(\alpha + \beta) \gamma = \alpha \gamma + \beta \gamma$

$\phi \alpha = \alpha \phi = \phi$

$(\alpha *) * = \alpha *$

$\alpha + \phi = \alpha$

1.5 რეგულარულ სიმრავლეებსა და მარჯვნივწრფივ გრამატიკებს შორის კავშირი

ვთქვათ, მოცემულია გრამატიკა G=(N , $\sum$, P , S). თუ ჩვენ P-ს წესებს დავადებთ გარკვეულ შეზღუდვებს, მივიღებთ სხვადასხვა ტიპის გრამატიკებს, რომლებიც ფართოდ ცნობილია ფორმალურ გრამატიკათა თეორიაში და შემოტანილია ხომსკის მიერ. გრამატიკას ეწოდება მარჯვნივ წრფივი, თუ მის ყოველ წესს აქვს სახე:

$A \to XB$ ან $A \to X$, სადაც $A,B \in N$ და $X \in \sum^*$.

გრამატიკას ეწოდება კონტექსტისაგან თავისუფალი ან კონტექსტისადმი არამგრ'ნობიარე, თუ მის ყოველ წესს აქვს სახე: $A \to \alpha$, სადაც $A \in N$, $\alpha \in (N \cup \sum)^*$. გრამატიკას ეწოდება კონტექსტზე დამოკიდებული ან კონტექსტისადმი მგრ'ნობიარე, ზოგჯერ არადამოკლებადი თუ მის ყოველ წესს აქვს სახე $\alpha \to \beta$, სადაც $|\alpha| \leq |\beta|$.

თვითეული ტიპის გრამატიკა განსაზღვრავს გარკვეული კლასის ენებს, რომელთაც გარკვეული თვისებები გააჩნიათ. საზოგადოდ შევნიშნოთ, რომ კონკრეტული L ენა განისაზღვრება ცალსახად რაიმე გრამატიკით, ე. ი. თუ L ენა განისაზღვრება G გრამატიკით ეს იმას არ ნიშნავს, რომ არ არსებობს რაიმე G_1 გრამატიკა G გრამატიკისაგან განსხვავებული, რომელიც იგივე L ენას განსაზღვრავს. ან სხვანაირად, თუ $L(G_1)= L(G_2)$, მიუხედავად ამისა თუ L ენა განისაზღვრება რაიმე ტიპის გრამატიკით, ჩვენ შეგვი'ლია ვილაპარაკოთ L ენის გარკვეულ თვისებებზე. დავიწყოთ უმარტივესი გრამატიკებით- მარჯვნივ წრფივი გრამატიკებით და ვაჩვენოთ, რომ მათ მიერ განსაზღვრული ენები არიან რეგულარული სიმრავლეები და პირიქით ნებისმიერი რეგულარული სიმრავლე განისაზღვრება რაიმე მარჯვნივ წრფივი გრამატიკით.

არსებობს მჭიდრო კავშირი რეგულარულ სიმრავლეებსა და მარჯვნივწრფივ გრამატიკებს შორის. სამართლიანია დებულება, რომ ნებისმიერი P რეგულარული სიმრავლისათვის შეიძლება აიგოს შესატყვისი მარჯვნივწრფივი გრამატიკა G ისეთი, რომ L(G)=P და პირიქით, ნებისმიერი G მარჯვნივწრფივი გრამატიკით განსაზღვრული ენა L(G) არის რეგულარული სიმრავლე. ამ დებულების სრული დამტკიცება იხილეთ [1]-ში. აქ, ჩვენ მხოლოდ ამ დებულების პირველ ნაწილს განვიხილავთ, რომელიც ხშირად გამოიყენება პრაქტიკაში მოცემული P რეგულარული სიმრავლით G მარჯვნივწრფივი გრამატიკის ასაგებად, რომლისთვისაც L(G)=P. განვიხილოთ რეგულარული სიმრავლეები: ϕ, {e} და {a} $a \in \sum$ და ავაგოთ მათთვის მარჯვნივწრფივი გრამატიკები, რომლებიც ამ სიმრავლეებს განსაზღვრავენ.

1. განვიხილოთ მარჯვნივწრფივი გრამატიკა

$G = (\{S\}, \sum, \phi, S)$ ცხადია $L(G)=\phi$.

2. განვიხილოთ G=({S, Σ ,{S$\rightarrow$ e},S) . ეს გრამატიკა მარჯვნივწრფივია და ცხადია L(G)={e}.

3. განვიხილოთ G=({S}, Σ ,{S$\rightarrow$ a},S). იგულისხმება a$\in$ Σ . ცხადია L(G)={a} და G მარჯვნივწრფივი გრამატიკაა.

ახლა ვაჩვენოთ თუ L_1 და L_2 მარჯვნივწრფივი ენებია, მაშინ:

1. L_1*L_2;

2. L_1L_2;

3. L_1*

მარჯვნივწრფივი ენებია.

1. ვთქვათ, L_1 მარჯვნივწრფივ ენას განსაზღვრავს. მარჯვნივწრფივი გრამატიკა G1=(N_1,Σ,P_2,S_1) და L_2 მარჯვნივწრფივ ენას განსაზღვრავს G2=(N_2, Σ , P_2,S_2). ავაგოთ ახალი მარჯვნივწრფივი გრამატიკა G3=($N_1$$\cup$$N_2$$\cup${S3},$P_1$$\cup$$P_2${$S_3$$\rightarrow$$S_1$| S2},$S_3$). ვაჩვენოთ, რომ L($G_3$)=L($G_1$)$\cup$L($G_2$). ყოველი $S_3$$\Rightarrow$*W-სათვის გრამატიკაში G_3 არსებობს გამოყვანა $S_1$$\Rightarrow$ *W G_1-ში და $S_2$$\Rightarrow$ *W G_2 –ში ე. ი. W$\in$ $L_1$$\cup$ L_2.. რადგანაც S_3 $\Rightarrow$ *W C_3 ში ნიშნავს S_3 $\Rightarrow$ S_1 $\Rightarrow$ *W ან S_3 $\Rightarrow$ S_2 $\Rightarrow$ *W G3-ში.

2. ვთქვათ G_3=(N_1 $\cup$ N_2, Σ ,P_3,S1) მარჯვნივწრფივი გრამატიკაა, სადაც P_3 განსაზღვრულია შემდეგნაირად:

a). თუ A$\rightarrow$ xB ეკუთვნის P_1-ს, მაშინ იგი ეკუთვნის ასევე P_3-ს. b).თუ A $\rightarrow$ X ეკუთვნის P_1–ს, მაშინ A$\rightarrow$ xS_2 ეკუთვნის P_3-ს. c) ყველა წესები P_2 –დან ეკუთვნის P_3-ს.

ცხადია, რომ თუ S_1 $\Rightarrow$ $^+$W G_1 –ში, მაშინ S_1 $\Rightarrow$ $^+$WS_2 G_3 –ში. თუ $S_2$$\Rightarrow$ $^+$V G_2-ში, მაშინ $S_1$$\Rightarrow$ $^+$WV G_3-ში. ე.ი. L(G_1) L (G_2) $\subseteq$ L(G_3).

ახლა, ვთქვათ $S_1$$\Rightarrow$$^+$W G_3-ში. რადგან G_3-ში არა გვაქვს A$\rightarrow$ X წესი, ამიტომ S_1 $\Rightarrow^+$ xS_2 $\Rightarrow^+$ xy=W, სადაც x$\in$ L_1 და y$\in$ L_2. ამგვარად, L(G_3) L(G_1)L(G_2) და საბოლოოდ L(G_3)=L(G_1)L(G_2). 3. ვთქვათ G_a=(N_1 $\cup$ S_a, Σ , P_a,S_a), სადაც S_a არ ეკუთვნის N_1- ს და P_a აგებულია შემდეგნაირად:

თუ A $\rightarrow$ xB ეკუთვნის P_1-ს, მაშინ იგი ეკუთვნის ასევე P_a-ს.

თუ A$\rightarrow$ x ეკუთვნის P_1-ს, მაშინ A$\rightarrow$ X S_a და A$\rightarrow$ X ეკუთვნიან P_a– ს.

ცხადია, $S_a \Rightarrow {}^{+}x_1S_a \Rightarrow {}^{+}x_1x_2S_a \Rightarrow {}^{+}\ldots \Rightarrow {}^{+}x_1x_2\ldots x_n\ S_a \Rightarrow x_1x_2\ldots x_n$ მიიღება G_a-ში მაშინ და მხოლოდ მაშინ, როცა $S_1 \Rightarrow {}^{+}x_1$, $S_1 \Rightarrow {}^{+}x_2,\ldots, S_1 \Rightarrow +x_n$ G_1-ში. ეს კი ნიშნავს, რომ $L(G_a)=(L(G_1))^*$.

1.6 სასრული ავტომატები

სასრული ავტომატი წარმოადგენს რეგულარული სიმრავლის წარმოდგენის ერთ-ერთ საშუალებას. მისი საშუალებით ჩვენ შეგვიძლია იოლად წარმოვადგინოთ ნებისმიერი რეგულარული სიმრავლე კომპიუტერზე. სასრული ავტომატი ჩვენ შეგვიძლია ავაგოთ როგორც მოწყობილობა, რომელსაც აქვს შესასვლელი ფირი და მმართველი მოწყობილობა სასრული მეხსიერებით. ფირზე მოდებულია თავაკი, რომელსაც შეუძლია წაიკითხოს ფირიდან მოცემულ მომენტში ერთი უჯრედი და თავაკს შეუძლია გადაადგილება ერთი უჯრედიდან მეორეზე. შესასვლელი ფირი დაყოფილია უჯრედებად და თვითეულ უჯრედში შეიძლება იყოს ჩაწერილი ალფაბეტის ერთი რომელიმე სიმბოლო. სიმარტივისათვის ჩვენ ვიგულისხმებთ, რომ თავაკს შეუძლია მოძრაობა ფირზე მარცხნიდან მარჯვნივ მიმართულებით. ასევე ვიგულისხმოთ, რომ ფირს აქვს უსასრულო სიგრძე, სადაც სიგრძის ქვეშ იგულისხმება უჯრედების რაოდენობა ფირზე. მმართველი მოწყობილობა შედგება მდგომარეობათა სასრული რაოდენობისაგან. მდგომარეობას ჩვენ ვუწოდებთ კვანძს და ჩვენი მოწყობილობა მოცემულ მომენტში შეიძლება იმყოფებოდეს ერთ რომელიმე კვანძში და ამ მოწყობილობას შეუძლია გადაადგილდეს ერთი კვანძიდან მეორეში იმისდამიხედვით თუ რა სიმბოლოა მოდებული თავაკზე. თავდაპირველად ვიგულისხმოთ, რომ მოწყობილობა იმყოფება რაიმე კვანძში და მას ვუწოდოთ საწყისი კვანძი (მდგომარება). ასევე მოწყობილობაში მონიშნულია რაიმე კვანძების ქვესიმრავლე, რომლებსაც ეწოდება საბოლოო მდგომარეობები. ასეთ მოწყობილობას ეწოდება სასრული ავტომატი. უჯრედში შეიძლება ჩაწერილი იყოს რაიმე სიმბოლო, წინააღმდეგ შემთხვევაში ასეთ უჯრედს ეწოდება ცარიელი უჯრედი. ცარიელი უჯრედის შემთხვევაში ჩვენ ვამბობთ, რომ მასში ჩაწერილია ცარიელი სიმბოლო. სიმბოლოების სიმრავლეს, რომლებიც შეიძლება შეგვხვდნენ ფირის უჯრედებში ვუწოდებთ ალფაბეტს. ცარიელი სიმბოლო არ ეკუთვნის ალფაბეტს.სიმბოლოების სასრულ მიმდევრობას ვუწოდოთ სიტყვა. ჩვენ ვიტყვით, რომ რაიმე სიტყვა მოდებულია სასრულ ავტომატზე, თუ ავტომატი იმყოფება საწყის მდგომარეობაში და თავაკი უთითებს სიტყვის პირველ სიმბოლოს ფირზე. სასრული ავტომატი გრაფიკულად ჩვენ შეგვიძლია

წარმოვიდგინოთ შემდეგნაირად: სასრული ავტომატის მდგომარეობები აღვნიშნოთ წერტილებით, რომლებიც მონიშნულია $q_i,\ldots,q_n$ ნიშნებით. ეს ნიშნები არ ეკუთვნიან ალფაბეტს. q_i მდგომარეობა შევაერთოთ q_j მდგომარეობასთან რკალით, რომელიც მონიშნულია რაიმე a_k სიმბოლოთი $a_k \in \Sigma$ (Σ აღნიშნავს ალფაბეტს) და რკალი მთავრდება ისრით, რომელიც მიმართულია q_i მდგომარეობიდან q_j მდგომარეობისკენ, თავაკი უთითებს a_k სიმბოლოს, მაშინ სასრულ ავტომატს შეუძლია გადავიდეს q_j მდგომარეობაში და თავაკი გადაადგილდება ერთი უჯრედით მარჯვნივ. წყვილს (q_j, W) ვუწოდოთ სასრული ავტომატის კონფიგურაცია, სადაც q_j მიმდინარე მდგომარეობაა და W არის სტრიქონი, რომელიც სასრულ ავტომატს ჯერ არ წაუკითხავს ფირიდან, ე. ი. ფირზე ჩაწერილია $W=a_{i1}\ldots a_{ik}$ სტრიქონი და თავაკი უთითებს a_{i1}–ს. ვთქვათ q_1 სასრული ავტომატის საწყისი მდგომარეობაა, $W_1=a_1\ldots a_n$ ფირზე ჩაწერილი სტრიქონია და სასრული ავტომატი საწყის მდგომარეობაში უთითებს a_1 სიმბოლოს, მაშინ საწყისი კონფიგურაცია იქნება (q_1,W_1). განვიხილოთ კონფიგურაციათა შემდეგი მიმდევრობა $(q_1,W_1), (q_2,W_2),\ldots,(q_n,W_n),(q_{n+1},e)$, სადაც $W_i=a_i\ldots a_n$ $(i=1,\ldots,n)$. თუ q_{n+1} საბოლოო მდგომარეობაა, მაშინ ჩვენ ვიტყვით, რომ სასრული ავტომატი უშვებს W_1 სტრიქონს. განვიხილოთ დამოკიდებულება $(q_i,W_i) \vdash (q_{i+1},W_{i+1})$, სადაც $(I=1,2,\ldots,n)$ და $W_{n+1}=e$. ჩანაწერი $(q_i,W_i) \vdash (q_{i+1},W_{i+1})$ აღნიშნავს სასრული ავტომატის q_i მდგომარეობიდან q_{i+1} მდგომარეობაში გადასვლას a_{i1} სიმბოლოთი. ეს იმას ნიშნავს, რომ სასრული ავტომატის დიაგრამაზე q_i კვანძი შეერთებულია q_{i+1} კვანძთან რკალით ისრით q_{i+1} –კენ და ეს რკალი მონიშნულია a_i სიმბოლოთი. ჩანაწერი $(q_1,W_1) \vdash^*(q_{n+1},e)$ აღნიშნავს შემოტანილი დამოკიდებულების ტრანზიტულ-რეფლექსურ ჩაკეტვას. გაშლილი სახით იგი ასე ჩაიწერება: $(q_1,W_1) \vdash (q_2,W_2) \vdash \ldots \vdash (q_n,W_n) \vdash (q_{n+1},e)$. თუ q_1 საწყისი მდგომარეობაა და q_{n+1} საბოლოო მდგომარეობა, მაშინ ვიტყვით, რომ $W_1=a_1\ldots a_n$ სტრიქონს უშვებს მოცემული სასრული ავტომატი, ან რაც იგივეა $W_1 \in L(A)$, სადაც A მოცემული ავტომატია და $L(A)$ არის ამ ავტომატით განსაზღვრული ენა. ყველა ასეთი W_1-ები ქმნიან $L(A)$ ენას $L(A)=\{W_1 | (q_1,W_1) \vdash^*(q_{n+1},e)\}$.

განვიხილოთ მაგალითი. ვთქვათ, მოცემულია სასრული ავტომატის დიაგრამა:

1 დიაგრამა

ისრით მითითებულია საწყისი მდგომარეობა, ხოლო კვანძები, რომლებიც აღნიშნულია O წრეებით, არიან საბოლოო მდგომარეობები. მაშასადამე, q_1

საწყისი მდგომარეობაა, ხოლო q_6, q_7 და q_{10} საბოლოო მდგომარეობები. თუ ამ დიაგრამით განსაზღვრულ ავტომატს აღვნიშნავთ A-თი, მაშინ L(A)-{სახლი, სახლმა, სახლს, სახე, სახემ^n, სახეს^n, სარი}. აქ n არის ნებისმიერი ნატურალური რიცხვი. შევნიშნოთ, რომ q_7 მდგომარეობიდან ს და მ სიმბოლოებით ავტომატი ისევ q_7 მდგომარეობაში რჩება, ხოლო q_2 მდგომარეობიდან S სიმბოლოთი სასრულ ავტომატს შეუძლია გადავიდეს q_3 მდგომარეობაში ან q_8 მდგომარეობაში. ამიტომ, ეს ავტომატი უშვებს სახემ, სახემმ და ა. შ. სტრიქონებს. შემოვიტანოთ განმარტება: თუ სასრულ ავტომატს რაიმე მდგომარეობიდან ერთი და იგივე სიმბოლოთი შეუძლია ერთზე მეტ მდგომარეობაში გადასვლა, მაშინ ასეთ სასრულ ავტომატს ეწოდება არადეტერმინისტული, წინააღმდეგ შემთხვევაში დეტერმინისტული. დიაგრამაზე გამოსული სასრული ავტომატი არადეტერმინისტულია.

ამის შემდეგ, შემოვიტანოთ სასრული ავტომატის მკაცრი განმარტება. A = (Q, Σ , δ , q_0, F) არის არადეტერმინისტული სასრული ავტომატი, სადაც

Q არის მდგომარეობათა სასრული სიმრავლე,

Σ - შესასვლელ სიმბოლოთა სასრული სიმრავლეა,

q_0 – საწყისი მდგომარეობაა და $q_{0\in}$ Q ,

F – საბოლოო მდგომარეობათა სიმრავლეა და F $\subseteq$ Q, არის Q X Σ დეკარტული ნამრავლის გადასახვა (Q) –ში ე. ი. სიმრავლეში, რომლის ელემენტებია Q – ს ქვესიმრავლეები.

მოცემული განმარტებით დიაგრამა 1 წარმოიდგინება ასე:

A=({q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8,q_9,q_{10},q_{11}}, {ს,ა,რ,ხ,ი,მ,ლ,ე}, δ, q_1, {q_6,q_7,q_{10}}), სადაც

δ (q_1, ს)={q_2}, δ (q_2, ა)={q_3,q_8},

δ (q_3,ხ)={q_4}, δ (q_4,ლ)={q_5}, δ (q_5,ი)={q_6}

δ (q_5,მ)={q_{11}}, δ (q_{11},ა)={q_6}, δ (q_4,ე)={q_7)

δ (q_7,მ)={q_7}, δ (q_7, ს)={q_7}, δ (q_8,რ)={q_9} და

δ (q_9,ი)={q_{10}}.

A სასრული ავტომატის არადეტერმინისტულობაზე მიუთითებს ის, რომ δ (q_2,a) სიმრავლე შეიცავს ორ ელემენტს. ახლა ჩვენ შეგვიძლია A სასრული ავტომატის კონფიგურაცია განვმარტოთ ასე: (q,W)∈ Q X Σ *, საწყისი კონფიგურაცია იქნება (q_0,W) და დამამთავრებელი კონფიგურაცია იქნება (q,e), სადაც q∈ F. ბინალური დამოკიდებულება ├ განისაზღვრება ასე: თუ δ (q,a) სიმრავლე შეიცავს q_1 –ს, მაშინ (q,aW). ეს ნიშნავს იმას, რომ თუ A სასრული ავტომტი იმყოფება q მდგომარეობაში და თავაკი მიუთითებს უჯრედს, რომელშიც ჩაწერილია a სიმბოლო, მაშინ სასრული ავტომატი გადადის q_1 მდგომარეობაში და თავაკი გადაადგილდება ერთი უჯრედით მარჯვნივ. ჩვენ ვიტყვით, რომ A სასრული ავტომატი უშვებს W სტრიქონს, როცა (q0,W) ├*(qe) და q∈ F. ასეთ შემთხვევაში W∈ L(A).

1.7 სასრული ავტომატის ცხრილური წარმოდგენა

ჩვენ შეგვიძლია სასრული ავტომატი აღებული წინა მაგალითიდან გამოვსახოთ შემდეგი ცხრილის სახით:

შესასვლელი								
ს	ა	რ	ხ	ი	მ	ლ	ე	
{q_1}	{q_2}							
q_2		{q_3,q_8}						
q_3				{q_4}				
q_4							{q_5}	{q_7}
q_5					{q_6}	{q_{11}}		
q_6						{q_7}		

q_7	$\{q_7\}$							
q_8			$\{q_9\}$					
q_9					$\{q_{10}\}$			
q_{10}								
q_{11}		$\{q_6\}$						

1 ცხრილი

ეს არის A სასრული ავტომატის ცხრილური წარმოდგენა. დიაგრამა 1-ზე წარმოდგენილი A სასრული დეტერმინისტული ავტომატი გადავაქციოთ სასრულ დეტერმინისტულ ავტომატად A_1 ისე, რომ $L(A)=L(A_1)$. ამისათვის საჭიროა, რომ q_8 დავამთხვიოთ q_3 – ს და q_3-დან უნდა გამოდიოდეს ის რკალები, რომლებიც გამოდიოდა q_8-დან. მის შემდეგ q_8- საერთოდ ამოვარდება მდგომარეობათა სიმრავლიდან. ასეთნაირად მიღებული ახალი სასრული ავტომატი A_1იქნება დეტერმინირებული და $L(A)=L(A_1)$.საზოგადოდ, მტკიცდება თეორემა, რომელიც ამბობს, რომ ნებისმიერი არატერმინირებული A სასრული ავტომატი შეიძლება გარდაიქმნას ახალ დეტერმნირებულ სასრულ ავტომატად ისე, რომ $L(A)=L(A_1)$.

დამტკიცება იხილეთ [1]-ში. რადგანაც დეტერმინირებული სასრული ავტომატის შემთხვევაში $\delta(q,a)$ სიმრავლე, ყოველი $a\in\sum$ და $q\in Q$-სთვის, არ შეიცავს ერთზე მეტ ელემენტს, ამიტომ ნაცვლად $\delta(q,a)=\{p\}$ შეგვიძლია შემოკლებით დავწეროთ $\delta(q,a)=P$. და როცა $\delta(q,a)=\phi$ (ცარიელი სიმრავლეა) მაშინ ვიტყვით, რომ $\delta(q,a)$ განუსაზღვრელია.

შეიძლება მარტივად დამტკიცდეს შემდეგი

დებულება: თუ რაიმე დეტერმინირებული სასრული ავტომატი M უშვებს L(M) ენას, მაშინ არსებობს ისეთი მარჯვნივწრფივი გრამატიკა G, რომ L(G)=L(M).

ვთქვათ $M=(Q,\sum,\delta,q_0,F)$ და $G=(Q,\sum,P,q_0)$ სადაც P განისაზღვრება შემდეგნაირად: თუ $\delta(q,a)=r$, მაშინ P შეიცავს წესს $q\rightarrow ar$ და თუ $q_1\in F$, მაშინ $q\rightarrow e$ ეკუთვნის P-ს. ასეთნაირად აგებული G გრამატიკა უშვებს იგივე ენას, რასაც M დეტერმინისტული სასრული ავტომატი, რის დამტკიცებაც ადვილია. პირიქით ზემოთ მოყვანილი წესების შებრუნებით გამოყენებით მარჯვნივწრფივი გრამატიკისათვის შეიძლება აიგოს დეტერმინირებული სასრული ავტომატი, რომელიც განსაზღვრავს იგივე ენას რასაც მოცემული მარჯვნივწრფივი გრამატიკა. ასევე შეიძლება ნებისმიერი რეგულარული სიმრავლისათვის აიგოს ისეთი დეტერმინისტული სასრული ავტომატი, რომელიც უშვებს მოცემულ

რეგულარულ სიმრავლეს. აქედან გამომდინარეობს, რომ სასრული ავტომატებითა და მარჯვნივწრფივი გრამატიკებით განსაზღვრული ენები არიან რეგულარული სიმრავლეები. ვისაც აინტერესებს სასრული ავტომატების მინიმიზაციის საკითხი (გარკვეული აზრით) და რეგულარულ სიმრავლეებთან დაკავშირებული გადაწყვეტადი პრობლემები ვურჩევთ [1]-ს.

1.8 რეგულარული გამოსახულება "Perl"-ში

რეგულარული გამოსახულებები ფართოდ გამოიყენებიან Perl ენაზე დაწერილ პროგრამებში. ამ თავში ჩვენ აღვწერთ რეგულარულ გამოსახულებებს Perl –ის მიხედვით. რეგულარული გამოსახულებების შემოტანის წყალობით Perl არის ბუნებრივი ტექსტების დამუშავების მძლავრი საშუალება. აქამდე აღწერილი რეგულარული გამოსახულებები მნიშვნელოვნად არის გაფართოებული Perl-ში. Perl-ის რეგულარული გამოსახულება არის რაიმე სტრიქონი, რომელიც აღწერს რაიმე ნიმუშს. ნიმუშები გამოიყენებიან ტექსტში კონკრეტული სტრიქონის მოსაძებნად, მის შესაცვლელად სხვა სტრიქონით, ტექსტიდან მისი ამოსაგდებისათვის, ან ტექსტის უფრო რთული გარდაქმნისათვის.

ჩვენ დავიწყებთ უმარტივესი რეგულარული გამოსახულების განსაზღვრით, ვაჩვენებთ მის გამოყენებას და შემდეგში თანდათანობით გავაფართოებთ ამ ცნებას. უმარტივესი რეგულარული გამოსახულებაა ნიშნების რაიმე მიმდევრობა რაიმე წინასწარ მოცემულ ალფაბეტში. რეგულარული გამოსახულება შემოსაზღვრული დახრილი ხაზებით (/). მაგალითად, /ab1/ არის ნიმუში, რომელიც შეიცავს ab1 რეგულარულ გამოსახულებას ASCII სტანდარტით განსაზღვრულ ალფაბეტში. იმისათვის, რომ განვსაზღვროთ რაიმე ნიმუში ეთანადება თუ არა მოცემულ ტექსტს, ამისათვის გამოიყენება ოპერატორი = ~ , რომლის მარჯვენა მხარეში იწერება ნიმუში, ხოლო მარცხენა მხარეში იწერება ტექსტი ბრჭყალებში ან ცვლადი, რომელიც შეიცავს ტექსტს. = ~ ოპერატორი წარმატებით სრულდება ან რაც იგივეა ეთანადება შაბლონს თუ შაბლონით განსაზღვრული რეგულარული გამოსახულება გვხვდება ამ ოპერატორის მარცხენა მხარით განსაზღვრულ ტექსტში, წინააღმდეგ შემთხვევაში ოპერატორი განიცდის მარცხს (შაბლონი არ ეთანადება ტექსტს). მაგალითად, ინსტრუქცია

″ this is a text ″ = ~ /is/;

წარმატებით სრულდება, რადგანაც is რეგულარული გამოსახულება პირველად გვხვდება this სიტყვაში. ოპერატორი ! ~ წარმატებით სრულდება თუ მის მარჯვენა მხარეში მოთავსებული გამოსახულება არ გვხვდება მარცხენა მხარეში.

მაგალითად, ″ this is a text ″ = ~ /at / არ ეთანადება;
″ this is atext ″ = ~ /at / ეთანადება.

ალფაბეტის ნიშნები რეგულარულ გამოსახულებაში იწერება ისე, როგორც ისინი გვხვდება ალფაბეტში, გამონაკლისს წარმოადგენენ ზოგიერთი ნიშნები, რომლებიც გამოიყენებიან რეგულარულ გამოსახულებაში სპეციალური დანიშნულებით. ასეთ ნიშნებს ეწოდებათ მეტანიშნები. მეტანიშნებია: [ ] ( ) { } ∧ + ∗ ?. იმისათვის რომ რაიმე მეტანიშანი გაგებული იყოს როგორც ჩვეულებრივი ნიშანი, მას წინ უნდა დაეწეროთ \ (შებრუნებული დახრილი ხაზი). მაგალითად \[გაიგება როგორც გახსნილი კვადრატული ფრჩხილი ან \+ გაიგება როგორც + ნიშანი და ა. შ.

მაგალითები: ″ 3 + 5 = 8 ″ = ~ / 3 + 5 / არ ეთანადება.

″ 3 + 5 = 8 ″ = ~ / 3 \ + 5 / ეთანადება.

″ a . b ∗ c ″ = ~ / a . b ∗ c / არ ეთანადება.

″ a . b ∗ c ″ = ~ / a \ . b \ ∗ c / ეთანადება.

\\ რეგულარულ გამოსახულებაში აღნიშნავს \ (შებრუნებულ დახრილ ხაზს). ASCI I ნიშანთა სიმრავლის ნიშნები, რომელთა კოდებია 0÷ 31, აქვთ სპეციალური აღნიშვნები. მაგალითად, \n აღნიშნავს ახალ სტრიქონზე გადასვლას, \t აღნიშნავს ტაბულაციას და სხვა. ეს აღნიშვნები შეიძლება გამოყენებულ იქნეს ასევე რეგულარულ გამოსახულებებშიც. ნიშნის ნაცვლად შესაძლებელია გამოყენებულ იქნეს მისი კოდი თექვსმეტობით ან რვაობით სისტემაში. თექვსმეტობითი კოდი წარმოიდგინება \ox n_1n_2, სადაც n_1 და n_2 თექვსმეტობითი ციფრებია, ან \o n_1 n_2 n_3 როგორც რვაობითი კოდი, სადაც \ n_1 n_2 და n_3 რვაობითი ციფრებია, სადაც o არის ო ასო და არა ნული. მაგალითად \ ox f 5 ან \ o 125. თვით \ (შებრუნებული დახრილი ხაზი) არის მეტანიშანი და მის წარმოსადგენად, როგორც ჩვეულებრივი ნიშანი საჭიროებს ნიშანს - \. ასე, რომ \\ არის დახრილი ხაზის ჩვეულებრივ ნიშნად წარმოდგენა. მაგალითად, ‘a \ b’ = ~ / a \\ b / ეთანადება, ე.ი. წარმატებით შესრულდება. მაგალითად, მივიღოთ, რომ თარიღს წარმოვადგენთ ტექსტში, როგორც თვე: რიცხვი: წელი, სადაც თვისთვის და რიცხვისთვის გამოყოფილია ორი ათობითი ციფრი, ხოლო წლისათვის ოთხი ათობითი რიცხვი: მაგალითად, 09: 28: 1973 ნიშნავს 28 სექტემბერი 1973 წელი. ასეთ შემთხვევაში ტექსტში რომ ვიპოვოთ რაიმე თარიღი საჭიროა მისთვის შევადგინოთ შემდეგი ნიმუში: / dd: dd: dddd / სადაც d ნიშნავს ნებისმიერ ათობით ციფრს. იგულისხმება, რომ თვე მოთავსებულია დიაპაზონში 01-12 და თვე-01: 31. ეს, რომ გავითვალისწინოთ, მაშინ დაგვჭირდება უფრო რთული ნიმუშის შედგენა.

რეგულარულ გამოსახულებაში შეიძლება გვქონდეს ცვლადებიც. Parl-ში სკალარული ცვლადის სახელი იწყება $ ნიშნით. აგალითად, $ name და $ day. რიან ცვლადის სახელები. ცვლადისთვის მნიშვნელობის მინიჭება ხდება = ოპერატორით. მაგალითად, $ day=′ 15`; ტექსტური კონსტანტების წარმოსადგენად

გვაქვს ორი საშუალება: ტექსტის მოთავსება ერთმაგ ბრჭყალებში ან ორმაგ ბრჭყალებში. თუ ტექსტი მოთავსებულია ერთმაგ ბრჭყალებში, მაშინ იგი გაიგება, როგორც წარმოდგენილი ნიშნების უბრალო მიმდევრობა. მაგალითად, ‘ a \ . + 5`, ხოლო თუ იგი მოთავსებულია ორმაგ ბრჭყალებში, მაშინ მხედველობაში მიიღება მეტასიმბოლოების მნიშვნელობები და ცვლადების მნივნელობები. ასეთ შემთხვევაში ამბობენ, რომ ხდება ტექსტის ინტერპოლირება. მაგალითად ავიღოთ " a \ n b $day ″ . მისი ინტერპოლირების შედეგად მიიღება " a \ n 5 ″ თუ $ day=’ 15 `; და \ n გაიგება როგორც ერთი ნიშანი, რომელიც ნიშნავს ახალ სტრიქონზე გადასვლას, ხოლო ‘ a \ n b $day` ნიშნავს იმ ნიშნების მიმდევრობას, რომელიც მოცემულია ბრჭყალებში. თუ ჩვენ გვინდა რომ ორმაგ ბრჭყალებში მოცემული იგივე იყოს რაც ერთმაგ ბრჭყალებში მოცემული ტექსტი, მაშინ უნდა ჩავწეროთ ასე: ‘’a \\ n b \ $day’’. ვთქვათ, $ day = ‘ Sunday ` ; მაშინ

‘today is Sunday ` = ~ / $day / ეთანადება.

‘today is Sunday ` = ~ / is Sunday / ეთანადება.

‘today is Sunday ` = ~ / ${ day } ეთანადება.

მესამე შემთხვევა არის Perl –ის ხეშ მასივის ელემენტის სახით წარმოდგენა- ${ day }-ის მნიშვნელობაა ` Sunday ‘.თუ ჩვენ გვინდა, რომ რეგულარული გამოსახულება ემთხვეოდეს მოცემული ტექსტის ნაწილს დასაწყისიდანვე მაშინ რეგულარული გამოსახულება უნდა დავიწყოთ ნიშნით ∧, ხოლო თუ გვინდა, რომ მოცემული რეგულარული გამოსახულებით დამთავრდეს ტექსტი , მაშინ რეგულარულ გამოსახულებას ბოლოში უნდა დავუწეროთ $ ნიშანი. მაგალითად,

‘ regular day ` = ~ / ∧ regular / ეთანადება

‘ regular day ` = ~ / ∧ day / არ ეთანადება

‘ regular day ` = ~ / day $ / ეთანადება

‘ regular day ` = ~ / ∧ day / არ ეთანადება.

რეგულარულ გამოსახულებაში შეიძლება ნიშანთა კლასების გამოყენება, რაც მთელ რიგ შემთხვევებში საშუალებას გვაძლევს უფრო მოკლედ დაიწეროს რეგულარული გამოსახულება. კვადრატულ ფრჩხილებში მოთავსებული ნიშნების სიმრავლეს ეწოდება ნიშანთა კლასი. მაგალითად, [a b c] და იგი ნიშნავს ამ სიმრავლიდან აღებულ ნებისმიერ ნიშნის შეხვედრას ტექსტის მიმდინარე პოზიციაში. მაგალითად, / r [a b c] d / ეთანადება rad-ს, r b d ან r c d. თუ ჩვენ ავიღებთ / r [c b a] d / -ს მაშინ იგი ეთანადება r c d-ს, r b d ან rad. ე. ი.

იგივეს, მაგრამ განსხვავდება შედარების რიგით. ასევე, / [D d A a Y y] / ეთანადება day –ს რეგისტრის მიუხედავად: Day, dAy, DAY და ა.შ.., ე. ი. მათ ნებისმიერ კომბინაციას. [\$ a] კლასი შედგება ორი ნიშნისაგან $ a. [a b c d e] კლასი შეგვიძლია ჩავწეროთ შემოკლებით [a-e]. ეს იმ შემთხვევაში თუ გვაქვს მიმდევრობით ყველა ნიშნები კლასში. მაგალითად, [a-e g u-x] წარმოადგენს კლასს [a b c d e g u v x]. თუ კლასის ელემენტია ‘ – ‘ ნიშანი, მაშინ იგი უნდა ჩაიწეროს ბოლო ნიშნად. მაგალითად, [a – c] ნიშნავს [a b c] კლასს. კლასი ლათინური ალფაბეტისათვის ჩაიწერება ასე: [a – z a – z]. ნიშანთა ზოგიერთი კლასებისათვის გამოიყენება შემოკლებები: \ d აღნიშნავს ნებისმიერ ციფრს და წარმოადგენს [0 – 9] კლასს.

\ S არის ხარვეზის ნიშანთა კლასი და წარმოადგენს [\ \ t \ r \ f]

\ w არის ალფაციფრულ ნიშანთა კლასი და წარმოიდგინება [0-9a-zA-

z _]

\ D არის არა \d. ე. ი. ნებისმიერი ნიშანი გარდა ციფრისა, რაც ნიშნავს \d \ ნამრავლის დამატებას უნივერსალურ სიმრავლემდე.

\ S არის არა \s.

\ W არის არა \w.

. (წერტილი) ნიშნავს ნებისმიერ ნიშანს გარდა \ n. ეს შემოკლებები შეიძლება გამოვიყენოთ როგორც კლასის შიგნით ასევე მის გარეთ რეგულარულ გამოსახულებაში. მაგალითად [\ s \d] ნიშნავს ხარვეზის ნიშანთა კლასს ან ციფრს.

‘ ∧ ‘ ნიშანი კლასის შიგნით დასაწყისში ნიშნავს ამ კლასით წარმოდგენილი ნიშნების დამატებას უნივერსალურ სიმრავლემდე. ე. ი. ნებისმიერ ნიშანს, რომელიც განსხვავდება კლასში წარმოდგენილი ნიშნებისაგან. მაგალითად, [∧ a b c] არის ნებისმიერი ნიშანი, გარდა ′ a′ ,′ b ′ . ან ′ c ′- ი. [a b c ∧] არის ′ a ′ ,′ b ′ ,′ c ′ ან ′ ∧ ′ · ე. ი. ′ ∧ ′ კლასში დასაწყისის გარდა ნებისმიერ ადგილას ნიშნავს თავისთავს. მაგალითები:

/ \ w \ w \ w / -ეთანადება სამ ასოციფრულ ნიშანს.

/ . . / - ეთანადება ორ ნებისმიერ ნიშანს.

/ \ . / - ეთანადება წერტილს ′ . ′ .

/ [∧ a b] / - ეთანადება ნებისმიერ ნიშანს გარდა ′ a′ და ′ b′ -სი.

ზოგიერთი ნიშნები აღნიშნავენ ტექსტში გარკვეულ ადგილს და არ მოითხოვენ რაიმე ნიშანთან დამთხვევას. ასეთებს ჩვენ უკვე გავეცანით როგორიცაა ′ ∧ ′ და ′§′, რომლებიც მიუთითებენ სტრიქონის დასაწყისს და დამთავრებას შესაბამისად. კიდევ არის \ b ′ ნიშანი, რომელიც მიუთითებს ასოციფრული სიტყვის საზღვარს (დაწყებას ან დამთავრებას) , ე.ი. W კლასის დაწყებას ან დამთავრებას. მაგალითად, / \ b \ w / მიუთითებს ტექსტში ადგილს, რომლის შემდეგ გვაქვს ასოციფრული ნიშანი, რომლის უშუალოდ წინ არაა ასოციფრული ნიშანი. მაგალითები:

′ : - a b ′ = ~ / \ d \w / ეთანადება ′ a ′ - ს.

′′′′ = ~ /∧ $ / ეთანადება.

′′′′ = ~ /./ არ ეთანადება.

′′\n′′ = ~ /∧ .$ / არ ეთანადება.

′′a′′ = ~ / ∧ . $ / ეთანადება.

′′a \ n ′′ = ~ / ∧ .$ / ეთანადება.

მოდიფიკატორები. როცა რეგულარული გამოსახულება მოთავსებულია დახრილ ხაზებს შორის (/ /) და ბოლო დახრილ ხაზს არ მოსდევს რაიმე ნიშანი ჩვენ ვამბობთ, რომ ეს არის სტანდარტული ნიმუში და მას არ გააჩნია რაიმე მოდიფიკატორი. თუ // ხაზებს მოსდევს სპეციალური ნიშნები, რომლებსაც ჩვენ ქვემოთ ჩამოვთვლით, მაშინ ვამბობთ, რომ რეგულარული გამოსახულება მოდიფიცირებულია ე. ი. იგი იქცევა შეთანხმებისას სხვანაირად იმისდამიხედვით თუ რა ნიშნები მოსდევს.

//S ნიშნავს, რომ შესათანადებელი სტრიქონი განიხილება, როგორც ერთი გრძელი სტრიქონი და ′.′ ეთანადება ნებისმიერ ნიშანს გარდა ′\n′-ის წინ.

//m ნიშნავს, რომ გვაქვს რამოდენიმე გრძელი სტრიქონი, რომელიც შეიცავს მრავალ სტრიქონს. ასეთ შემთხვევაში ′ . ′ ეთანადება ნებისმიერ ნიშანს გარდა ′ \ n ′ -სა. ′ ∧ ′ და ′ $ ′ ნიშნებს შეუძლიათ შეუთანადდნენ ნებისმიერი სტრიქონის დასაწყისსა და დაბოლოებას შესაბამისად მთლიან გრძელ სტრიქონში.

\\sm ნიშნავს რომ ამუშავებს ერთ გრძელ სტრიქონს, მაგრამ მის შიგნით აღმოაჩენს მრავალ სტრიქონს. ′ . ′ ეთანადება ნებისმიერ ნიშანს ′ \ n ′ - საც. ′

^ ′ და $′ შეუძლიათ შეეთანადონ ნებისმიერი სტრიქონის დასაწყისსა და დაბოლოებას სტრიქონის შიგნით. მაგალითები:

ალტერნატივები. რეგულარულ გამოსახულებაში შეიძლება გამოვიყენოთ ალტერნატივები ე.ი. რეგულარული გამოსახულების შიგნით შეიძლება გვქონდეს რამოდენიმე რეგულარული გამოსახულება და სავალდებულოა მხოლოდ რომელიმე მათგანის შეთანადება ტექსტში. ალტერნატივის მაჩვენებელია ′|′ (ვერტიკალური ნიშანი). მაგალითად, თუ ჩვენ გვინდა ვიპოვოთ ტექსტში ′სავარძელი′ ან ′სკამი′ ჩვენ უნდა შევადგინოთ ნიმუში: /სავარძელი|სკამი/.

მაგალითად, /[d e f] / იგივეა რაც /d|e|f /.

′ სკამი′ = ~ /სკამი|ს|′სკ/ ეთანადება ′სკამი′

′სკამი′ = ~ / ს|სკ|სკამი| ეთანადება ′ს′ -ს:

შეჯგუფება. ჩვენ შეგვიძლია რეგულარული გამოსახულების ნაწილი შევაჯგუფოთ და განვიხილოთ როგორც ერთი ელემენტარული ერთეული ან რაც იგივეა განვიხილოთ იგი როგორც ქვეგამოსახულება. ეს საშუალებას გვაძლევს უფრო კომპაქტურად ჩავწეროთ რეგულარული გამოსახულება და თითოეული შეჯგუფებული ნაწილი დავამუშაოთ როგორც დამოუკიდებელი რეგულარული გამოსახულება. შეჯგუფება ხდება მრგვალი ფრჩხილების გამოყენებით. მაგალითად, თუ გვაქვს რეგულარული გამოსახულება /დედაენა\დედაკაცი/ ჩვენ შეგვიძლია უფრო კომპაქტურად ჩავწეროთ შეჯგუფებით:

/დედა(ენა|კაცი)/

მაგალითები:

/(ა|გა|ჩა)(მო\) /# ეთანადება ამო, ა, გამო, გა, ჩამო, ჩა ან არა აქვს ზმნისწინი.

/სახლ(ის|ით|ი|მა|ს|ად|ო)/# ეთანადება ნებისმიერ ბრუნვაში მხოლობით რიცხვში დასმულ სიტყვას ″სახლი″.

/ სახლ | ((ებ|) (ის| ით| ი| მა| ად| ო))| (ნ| თ) / # ეთანადება მხოლობითში ან მრავლობითში ნებისმიერ ბრუნვაში დასმულ სიტყვას ′ ′ სახლი ′ ′ .

/ აცა | აც| ცა| ც| ღა| ავე| ვე \ / # ეთანადება რომელიმე ნაწილაკს ან არა აქვს ნაწილაკი.

Perl-ში თვითეულ მრგვალ ფრჩხილებში მოთავსებულ გამოსახულებას შეესაბამება $ n ცვლადი, სადაც n ნატურალური რიცხვია. $ n –ში ინახება მე – n

–ე მრგვალ ფრჩხილებში მოთავსებული გამოსახულების შესატყვისი ტექსტი. მაგალითად, $1 იქნება პირველ მრგვალ ფრჩხილებში მოთავსებულ რეგულარულ გამოსახულებასთან შეთანადებული ნუმერაცია. იწყება ერთიდან და გადაინომრება მარცხნიდან მარჯვნივ გახსნილი ფრჩხილები. მაგალითად, შეთანადების ინსტრუქციაში

″სახლისათვის″ = ~ m / ∧ ((· W +) (ის) (თვის \) (აც\)) $ /

ფრჩხილები გადაინომრება ასე: / ∧ ($_1$($_2$ W+)$_2$ ($_3$ ის)$_3$ ($_4$ თვის)$_4$ ($_5$ აც |)$_5$)$_1$ $ /

და შეთანადების ოპერატორის შესრულების შემდეგ:

§1 → ″სახლისთვისაც″

§2 → ″სახლს″

§3 → ″ის″

§4 → ″თვის″

§5 → ″აც″

§n ცვლადების მნიშვნელობები შეიძლება გამოვიყენოთ სხვა ინსტრუქციებში რეგულარული გამოსახულების გარეთ და მათ ჩვენ არ შეგვიძლია მინიჭების ინსტრუქციით მივანიჭოთ ახალი მნიშვნელობები. თუ ჩვენ გვინდა §n-ის მნიშვნელობა გამოვიყენოთ იგივე რეგულარული გამოსახულების შიგნით ჩვენ უნდა მივუთითოთ იგი \ნ სახით, მხოლოდ მას შემდეგ რაც §n განისაზღვრება რეგულარულ გამოსახულებაში. მაგალითად:

″თავთავი″ = “მ / ∧ ((თავ) \ 2 ი) § / # ეთანადება და

§1 იქნება თავთავი,

§2 იქნება თავ

2 თავი

კონტექსტისაგან თავისუფალი გრამატიკის გარდაქმნები და ნორმალური ფორმები

2.1 სასრული გარდამქმნელები

ერთი ფორმალური ენიდან მეორეზე თარგმანი შეიძლება შევასრულოთ მოწყობილობით, რომელიც წარმოადგენს სასრული ავტომატის განზოგადოებას. კერძოდ, სასრულ ავტომატს თუ დავუმატებთ გამოსასვლელ ფირს, რომელზედაც სასრული ავტომატი ყოველი ტაქტის შესრულებისას გამოიტანს რაიმე გამოსასვლელ სიმბოლოს, ჩვენ მივიღებთ მოწყობილობას, რომლის გამოტანა იქნება თარგმანი იმ სიმბოლოთა მიმდევრობისა, რომლის შეტანა მოხდა შესასვლელი ფირიდან სასრული ავტომატის მუშაობის დროს. ამ შემთხვევაში გამოტანა ეს არის გამოსასვლელ ფირზე გამოტანილი სიმბოლოების მიმდევრობა. ასეთ მოწყობილობას ეწოდება სასრული გარდამქმნელი.

ფორმალურად სასრული გარდამქმნელი განისაზღვრება როგორც ექვსეული

$(Q, \Sigma, \Delta, \delta, q_0, F)$, სადაც

Q – არის მდგომარეობათა სიმრავლე,

Σ - შესასვლელ სიმბოლოთა სიმრავლე,

Δ - გამოსასვლელ სიმბოლოთა სიმრავლე,

q_0 - საწყისი მდგომარეობაა,

δ - არის $Q \times (\Sigma \cup \{e\})$ სიმრავლის გადასახვა $Q \times \Delta^*$ სიმრავლის ქვესიმრავლეთა სიმრავლეში.

სასრული ავტომატის ანალოგიურად განიმარტება სასრული გარდამქმნელის კონფიგურაცია. მხოლოდ აქ კონფიგურაციას დაემატება ახალი ელემენტი – გამოსასვლელი სტრიქონი. ამგვარად, სასრული გარდამქმნელის კონფიგურაციაა სამეული (q, x, y), სადაც q აღნიშნავს გარდამქმნელის მდგომარეობას, x – არის თავდაპირველი შესასვლელი სტრიქონის დარჩენილი ნაწილი მოცემული

მომენტისათვის, ე. ი. მისი სუფიქსი, ხოლო y არის მოცემული მომენტისათვის გამოტანილი სტრიქონი ანუ თავდაპირველი სტრიქონის პრეფიქსი.

სასრული გარდამქმნელი ყოველი ტაქტის შესრულებისას გადადის ერთი კონფიგურაციიდან მეორეში. დამოკიდებულება, რომელიც განსაზღვრავს სასრული გარდამქმნელის გადასვლას ერთი კონფიგურაციიდან მეორეში აღვნიშნოთ $\vdash$ -ით. ანალოგიურად, სასრული ავტომატისა, აქ ჩვენ შემოვიტანოთ ამ დამოკიდებულებისათვის ^{i}, $^{+}$ და * ოპერატორები. დამოკიდებულება განისაზღვრება ასე: $(q, a x, y) \vdash (r, x yz)$ სადაც

$q \in Q$ და $r \in Q$, $a \in (\Sigma \cup \{ e \}$, $x \in \Sigma^{*}$ და $y \in \nabla^{*}$, ხოლო $\delta (q, a)$ სიმრავლე უნდა შეიცავდეს (r, z) ელემენტს. ამის შემდეგ ჩვენ შეგვიძლია განვსაზღვროთ სასრული გარდამქმნელით მოცემული $\tau(M)$ თარგმანი:

$$\tau(M) = \{(x,y) \mid (q_0 ,x, e) \vdash^{*} (q,e,y) \text{ და } q \in F \}$$

(x, y) წყვილში იგულისხმება, რომ y არის x -ის თარგმანი. მიაქციეთ ყურადღება, რომ სასრული გარდამქმნელი ამთავრებს წარმატებით მუშაობას, როცა შესასვლელი სტრიქონი დაცარიელდება. მაგალითი:

$S \rightarrow a + S \mid a - S \mid + S \mid - \mid a$

$Q = \{ q_0,q_1,q_2,q_3,q_4 \}$ FF

$\Sigma = \{ a , + , - \}$

2.2 გამოყვანის ხეები

გრამატიკაში შეიძლება იყოს რამოდენიმე გამოყვანა ექვივალენტური იმ გაგებით რომ ყოველ მათგანში ერთი და იგივე გამოყვანის წესები გამოიყენება ერთი და იგივე ადგილებში,მაგრამ განსხვავებული რიგით. ექვივალენტობის ცნების განმარტება ორი გამოყვანისათვის ნებისმიერი სახის გრამატიკისათვის რთულია, მაგრამ CF გრამატიკისათვის შეიძლება შემოვიტანოთ ექვივალენტურ გამოყვანათა კლასის მოხერხებული გრაფიკული წარმოდგენა, რომელსაც ეწოდება გამოყვანის ხე. გამოყვანის ხე CF გრამატიკაში **G(N, Σ, P, S)** ეს არის დალაგებული ხე, რომლის ყოველი წვერო მონიშნულია რაიმე სიმბოლოთი $N \cup S \cup \{e\}$ სიმრავლიდან. თუ შიგა წვერო მონიშნულია A სიმბოლოთი, ხოლო მისი პირდაპირი შთამომავლები - $X_1X_2...X_n$ სიმბოლოებით, მაშინ $A \rightarrow X_1X_2...X_n$ არის ამ გრამატიკის წესი.

განმარტება. მონიშნულ დალაგებულ D ხეს ეწოდება გამოყვანის ხე (ან გარჩევის ხე) CF გრამატიკაში G(A)=(N,Σ,P,A) თუ შესრულებულია შემდეგი პირობები:

1. ხის ძირი მონიშნულია A სიმბოლოთი.

2. თუ $D_1,D_2,...,D_k$ D_i ხის ქვეხეებია და მონიშნულია X_i-თი, მაშინ $A \to X_1X_2...X_k$ წესია აღებული P სიმრავლიდან. D_i უნდა იყოს გამოყვანის ხე CF გრამატიკაში $G(X_i)=(N,\Sigma,P,X_i)$ თუ X_i არატერმინალია და D_i შედგება ერთადერთი წვეროსაგან, რომელიც მონიშნულია X_i-თი, თუ X_i ტერმინალია.

3. თუ ხის ძირს გააჩნია ერთადერთი შთამომავალი მონიშნული e-თი, მაშინ ეს შთამომავალი ქმნის ხეს, რომელიც შედგება ერთადერთი წვეროსაგან A და $A \to e$ წესია აღებული P სიმრავლიდან.

მაგალითი 2.2.1 ნახ.2.2.1-ზე გამოსახულია გამოყვანის ხეები G=G(S) გრამატიკაში წესებით

S →aSbS/bSaS/e.

S

ნახ.2.2.1 გამოყვანის ხეები.

შევნიშნოთ, რომ არსებობს დალაგებული ხის წვეროების ერთადერთი დალაგება,რომელშიც წვეროების პირდაპირი შთამომავლები დალაგებულია მარცხნიდან მარჯვნივ. განვმარტოთ ეს დალაგება შემდეგნაირად. დავუშვათ, რომ n არის წვერო და $n_1,n2,...,nk$ მისი პირდაპირი შთამომავლებია, მაშინ თუ $i<j$ n_{ij} წვერო და ყველა მისი შთამომავლები ითვლება განლაგებულად n_j წვეროს და მისი შთამომავლების მარცხნივ. იმის დამტკიცება, რომ ასეთი დალაგება არ შეიცავს წინააღმდეგობებს თვალსაჩინოა. საჭიროაბ მხოლოდ იმის ჩვენება, რომ ამ δ დალაგებული ხის ნებისმიერი ორი წვერო ძევს ერთსადაიგივე გზაზე, ან ერთ-ერთი მათგანი მოთავსებულია მეორის მარცხნივ.

განმარტება2.2.1. გამოყვანის ხის კრონი ვუწოდოთ ჯაჭვს, რომელიც მიიღება, თუ ამოვწერთ მარცხნიდან მარჯვნივ ფოთლების მონიშვნებს.

მაგალითად, ნახაზზე ნაჩვენები გამოყვანის ხეების კრონებია:

(a) S, (b) e, (c) abab, (d) abab.

ახლა ვაჩვენოთ, რომ გამოყვანის ხეები ადექვატურად წარმოადგენენ გამოყვანებს იმ აზრით, რომ CF გრამატიკაში G α გამოყვანილი ჯაჭვის ყოველი გამოყვანისათვის შეიძლება აიგოს გამოყვანის ხე G-ში α კრონით და პირიქით.ამისათვის,

შემოვიტანოთ რამოდენიმე ახალი ცნება. ვთქვათ, D არის გამოყვანის ხე CF გრამატიკაში G=(N,∑ ,P,S).

განმარტება2.2.2: D ხის კვეთა ვუწოდოთ D ხის ისეთ C წვეროების სიმრავლეს, რომ

1. არცერთი ორი წვერო C -დან არ ძევს ერთსა და იმავე გზაზე D-ში.

2. D ხის არცერთი წვეროს დამატება არ შეიძლება C სიმრავლისათვის, რომ არ დაირღვეს (1) თვისება.

მაგალითი2.2.1. ხის წვეროების სიმრავლე, რომლებიც შეიცავენ მხოლოდ ძირს არის კვეთა. ფოთლები აგრეთვე ქმნიან კვეთას. კვეთის კიდევ ერთ მაგალითს

ნახ.2.2.2. კვეთის მაგალითი.

აკლია სურათი 2.9

წარმოადგენს ნახ.-ზე მოცემული სიმრავლე,რომელიც შედგება წრეებში მოთავსებული წვეროებისაგან.

განმარტება2.2.3. განვმარტოთ D ხის კვეთის კრონი როგორც ჯაჭვი, რომელიც მიიღება მარცხნიდან მარჯვნივ იმ წვეროების მონიშვნებით კონკატენაციით,რომლებიც ქმნიან რომელიმე კვეთას. მაგალითად, aba SbS იმ კვეთის გამოყვანის ხის კრონია,რომელიც ნაჩვენებია სურათ 2.9-ზე.

ლემა2.2.1. $S=\alpha_0,\alpha_1,...,\alpha_n$ არის α_n ჯაჭვის გამოყვანა CF გრამატიკაში G=(N,∑ ,P,S). მაშინ G-ში შეიძლება აიგოს გამოყვანის ხე D, რომლისთვისაც α_n-კრონია,ხოლო $\alpha_0,\alpha_1,...\alpha_{n-1}$ რომელიმე კვეთების კრონებია. დამტკიცება. ავაგოთ $D_i(o\leq i\leq n)$ გამოყვანის ხეების ისეთი მიმდევრობა, რომ α_i იყოს კრონი D_i ხისა.ვთქვათ D_0 არის ხე, რომელიც შედგება ერთადერთი წვეროსაგან, რომელიც მონიშნულია S-ით. დავუშვათ რომ $\alpha_i=\beta_i A\gamma_i$ და $A\rightarrow X_1X_2...X_k$ წესის გამოყენების შემდეგ გამოყოფილ A ნაწილზე მიიღება $\alpha_{i+1}=\beta_i X_1X_2...X_k\gamma_i$ მაშინ D_{i+1} ხე მიიღება D_i-საგან თუ A-თი მონიშნულ ფოთოლზე (იგი არის ($|\beta_i|+1$) სიმბოლო D_i ხის კრონისა) დავუმატებთ K პირდაპირ შთამომავლებს და მათ მოვნიშნავთ $X_1,X_2,...,X_k$-თი შესაბამისად. ცხადია რომ D_{i+1} ხის კრონი იქნება α_{i+1}. D_n ხე იქნება საძიებელი გამოყვანის ხე. დავამტკიცოთ შესრულებული ლემა 2.12,ე.ი. ვაჩვენოთ, რომ ყოველი გამოყვანის ხისათვის G-ში არსებობს ერთი მაინც შესაბამისი გამოყვანა.

ლემა2.2.2. ვთქვათ D - გამოყვანის ხეა CF გრამატიკაში G=(N,∑ ,P,S) კრონით α .მაშინ $S\Rightarrow^*\alpha$.

დამტკიცება.ვთქვათ $C_0,C_1,...C_n$ D ხის ისეთი კვეთებია, რომ

1. C_0 შეიცავს მხოლოდ D ხის ძირს.

2. C_{i+1} ($0 \leq i < n$) მიიღება C_i_დან მასში ერთერთი არატერმინალური წვეროს შეცვლით მისი პირდაპირი შთამომავლებით.

3. C_n _ არის D ხის კრონი.

ცხადია, რომ ერთი მაინც ასეთი მიმდევრობა არსებობს. თუ α_i არის C_i_სი, მაშინ $\alpha_0,\alpha_1,\ldots,\alpha_n$ არის α_n ჯაჭვის გამოყვანა α_0-დან G-ში.

გამოყვანებს შორის, რომლებიც შეიძლება მოცემული ხისაგან აიგოს, ორი მათგანი ჩვენ განსაკუთრებით გვაინტერესებს.

განმარტება 2.2.4. თუ ლემის დამტკიცებაში C_{i+1} მიიღება C_i_საგან მისი ყველაზე მარცხენა არატერმინალური წვეროს შეცვლით მისი პირდაპირი შთამომავლებით, მაშინ სათანადო გამოყვანას $\alpha_0,\alpha_1,\ldots,\alpha_n$ ეწოდება G გრამატიკაში α_0-საგან α_n ჯაჭვის მარცხენა გამოყვანა. მარჯვენა გამოყვანა განიმარტება ანალოგიურად, საჭიროა მხოლოდ წინა წინადადებაში ყველაზე მარცხენა წვეროს ნაცვლად განვიხილოთ ყველაზე მარჯვენა წვერო. შევნიშნოთ რომ მარცხენა (ან მარჯვენა) გამოყვანა განისაზღვრება ცალსახად გამოყვანის ხით.

თუ $S=\alpha_0,\alpha_1,\ldots,\alpha_n=W$ არის W ჯაჭვის მარცხენა გამოყვანა, მაშინ α_i-ს($o \leq i < n$). აქვს სახე $X_iA_iB_i$ სადაც $X_i \in \Sigma^*$,$A_i \in N$ და $\beta_i \in (N \cup \Sigma)^*$ ყოველი შემდგომი მარცხენა გამოყვანის α_{i+1} ჯაჭვი მიიღება α_i-საგან მისი მარცხენა არატერმინალური A_i სიმბოლოს შეცვლით რომელიმე წესის მარჯვენა მხარით. მარჯვენა გამოყვანაში იცვლება მარჯვენა არატერმინალი.

მაგალითი2.2.2. განვიხილოთ CF G_0 გრამატიკა წესებით

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid a$

გამოყვანის ხე, რომელიც ნაჩვენებია ნახ-ზე, a+a ჯაჭვის 10 ექვივალენტური გამოყვანების წარმოდგენაა.

ნახაზი 2.11

მისი მარცხენა გამოყვანაა

$E \rightarrow E+T \rightarrow T+T \rightarrow F+T \rightarrow a+T \rightarrow a+F \rightarrow a+a$

ხოლო მარჯვენა გამოყვანაა

$E \rightarrow E+T \rightarrow E+F \rightarrow E+a \rightarrow T+a \rightarrow F+a \rightarrow a+a.$

განმარტება 2.2.5. α ჯაჭვს ვუწოდებთ მარცხნივ გამოყვანადს (G გრამატიკაში), თუ არსებობს $S=\alpha_0,\alpha_1,...,\alpha_n=\alpha$ მარცხენა გამოყვანა და დავწეროთ $S\Rightarrow^*_{GL\alpha}$ (ან $S\Rightarrow^*_{L\alpha}$, როცა ცხადია თუ რომელიმე G გრამატიკა იგულისხმება). ანალოგიურად α - სვუწოდებთ მარჯვნივ გამოყვანადს, თუ არსებობს მარჯვენა გამოყვანა $S=\alpha_0,\alpha_1,...,\alpha_n=\alpha$ და ვწერთ $S\Rightarrow_{Gr}^{*\alpha}$ (ან $S\Rightarrow_r^{*\alpha}$). მარცხენა გამოყვანის ერთ ნაბიჯს აღვნიშნავთ $\Rightarrow_L$, ხოლო მარჯვენა გამოყვანისას $\Rightarrow_r$-ით.1 და 2 ლემები შეიძლება გავაერთიანოთ ერთ თეორემად:

თეორემა 2.2.1. ვთქვათ $G=(N,\Sigma,P,S)$ CF გრამატიკაა. $S\Rightarrow^{*\alpha}$ მაშინ და მხოლოდ მაშინ, როცა G-ში არსებობს გამოყვანის ხე კრონით α .

დამტკიცება. ეს უშუალოდ გამომდინარეობს 2.2.1 და 2.2.2 ლემებიდან.

შევნიშნოთ, რომ ჩვენ თავი ავარიდეთ იმის თქმას,რომ თუ მოცემულია გამოყვანა $S\Rightarrow^{*\alpha}$ CF გრამატიკაში, მაშინ შეიძლება ვიპოვოთ G-ში ერთად-ერთი გამოყვანის ხე კრონით α . ამის მიზეზი მდგომარეობს იმაში, რომ არსებობენ CF გრამატიკები, რომელთაც შეიძლება ჰქონდეთ რამდენიმე განსხვავებული გამოყვანის ხეები ერთი და იმავე კრონით. ასეთია გრამატიკა მაგალითიდან. ნახ.-ზე ნაჩვენებია სხვადასხვა გამოყვანის ხეები (d) და (u) ერთიდაიგივე კრონით.

განმარტება 2.2.6. CF გრამატიკას G ეწოდება არაცალსახა თუ არსებობს ერთი მაინც $W\in L(G)$ ჯაჭვი, რომელიც არის ორი ან მეტი განსხვავებული გამოყვანის ხის კრონი G-ში. ეს იგივეა, რომ რომელიმე $W\in L(G)$ ჯაჭვს აქვს ორი ან მეტი მარცხენა (მარჯვენა) გამოყვანა,წინააღმდეგ შემთხვევაში CF გრამატიკას G-ს ცალსახა გრამატიკა ეწოდება.

2.3 კონტექსტისაგან თავისუფალ გრამატიკათა გარდაქმნები

ხშირად, საჭიროა მოცემული გრამატიკის ისეთი მოდიფიცირება, რომ მის მიერ წარმოქმნილმა ენამ მიიღოს საჭირო სტრუქტურა. განვიხილოთ, მაგალითად, $L(G_0)$ ენა. იგი შეიძლება მივიღოთ G გრამატიკის წესებით

$E\rightarrow E+T|\ E*T|\ T$

$T\rightarrow (E)|a$

G გრამატიკის მეორე ნაკლი, რომელიც გააჩნია აგრეთვე G_1 გრამატიკას, იმაში მდგომარეობს, რომ + და * ოპერაციებს აქვთ ერთი და იგივე პრიორიტეტი. სხვანაირად რომ ვთქვათ, a+a*a და a*a+a გამოსახულებების სტრუქტურა, რომელსაც ანიჭებს მათ G_1 გრამატიკა, გულისხმობს ოპერაციების შესრულების იგივე რიგს, რასაც გამოსახულებებში (a+a)*a დ ა (a*a)+a შესაბამისად.

რომ მივიღოთ + და * ოპერაციების ჩვეულებრივი პრიორიტეტი, რომლის დროსაც * წინ უსწრებს + ოპერაციას და გამოსახულება a+a*a გაიგება როგორც a+(a*a), საჭიროა გადავიდეთ G_0 გრამატიკაზე.

ზოგადი ალგორითმული მეთოდი, რომელიც მისცემდა მოცემულ ენას ნებისმიერ სტრუქტურას, არ არსებობს. მაგრამ, მთელი რიგი გარდაქმნების გამოყენებით შეიძლება შევუცვალოთ სახე გრამატიკას ისე, რომ არ შევცვალოთ მის მიერ წარმოქმნილი ენა. მოცემულ პარაგრაფში და 2.4.3-2.4.5 პარაგრაფებში ჩვენ მივუთითებთ ასეთი სახის რამოდენიმე გარდაქმნას.

დავიწყოთ ყველაზე ცხადით, მაგრამ მნიშვნელოვანი გარდაქმნებით. ზოგიერთ შემთხვევებში CF გრამატიკა შეიძლება შეიცავდეს უსარგებლო სიმბოლოებიან წესებს. მაგალითად, G=({S,A},{a,b},P,S) გრამატიკაში, სადაც P={S→a, A→b}, არატერმინალი A და b არ შეიძლება შეგვხვდეს არცერთ გამოყვანად ჯაჭვში. ამგვარად, ამ სიმბოლოებს არა აქვთ არავითარი დამოკიდებულება L(G) ენასთან და ისინი შეიძლება ამოვაგდოთ G გრამატიკის განსაზღვრიდან ისე, რომ არ შევეხოთ L(G) ენას.

განმარტება2.3.1. $X \in N \cup \Sigma$ სიმბოლოს ვუწოდოთ უსარგებლო CF გრამატიკაში $G=(N,\Sigma,P,S)$ თუ მასში არ არსებობს $S \Rightarrow {}^{*}WX\gamma$ სახის გამოყვანა,სადაც W,X,γ ეკუთვნიან Σ^{*}-ს.

იმისათვის, რომ დავადგინოთ A არატერმინალი უსარგებლოა თუ არა თავდაპირველად ავაგოთ ალგორითმი, რომელიც გაარკვევს შეიძლება თუ არა არატერმინალი ჰქონდეს რომელიმე ტერმინალურ ჯაჭვს, ე.ი. რომელიც ხსნის $\{W| A \rightarrow {}^{*}W, W \in \Sigma^{*}\}$ სიმრავლის სიცარიელის პრობლემას. ასეთი ალგორითმის არსებობიდან გამომდინარეობს CF გრამატიკისათვის სიცარიელის პრობლემის გადაწყვეტა.

ალგორითმი2.3.1.ცარიელია თუ არა L(G) ენა?

შესასვლელი. CF გრამატიკა $G=(N,\Sigma,P,S)$.

გამოსასვლელი. "დიახ" , თუ $L(G) \neq \varnothing$. " არა" წინააღმდეგ შემთხვევაში.

მეთოდი: ვაგებთ $N_0, N_1,...$ სიმრავლეებს რეკურსიულად:

1. დავუშვათ $N_0=\varnothing$ და i=1.

2. ავაგოთ $N_i=\{ A| A\rightarrow \alpha \in P$ და $\alpha \in (N_{i-1}\cup \sum)^*\} \} \cup N_{i-1}$

3. თუ N_{i-1}, მაშინ გავზარდოთ i ერთით და გადავიდეთ (2) ნაბიჯზე. წინააღმდეგ შემთხვევაში მივიღოთ $N_e=N_i$.

4. თუ $S\in N_e$, მაშინ გამოსასვლელზე გამოვიტანოთ ″ დიახ ″ , წინააღმდეგ შემთხვევაში ″ არა ″ . □

რადგან $N_e\subseteq N$, ამიტომ 2.7 ალგორითმი უნდა გაჩერდეს ყველაზე მეტი n+1-ჯერ (2) ნაბიჯის განმეორების შემდეგ, თუ N შეიცავს n რაოდენობა არატერმინალს. დავამტკიცოთ 2.7 ალგორითმის კორექტულობა. დამტკიცება მარტივია და შემდგომში გამოდგება როგორც მოდელი რამოდენიმე ანალოგიური დამტკიცებისათვის.

თეორემა 2.3.1. ალგორითმი ამბობს ″დიახ″ მხოლოდ და მხოლოდ მაშინ, როცა $S\Rightarrow {}^*W$ რომელიმე $W\in \sum^*$ ჯაჭვისათვის.

დამტკიცება. თავდაპირველად დავამტკიცოთ i-ს მიხედვით ინდუქციით, რომ

თუ $A\in N_i$, მაშინ $A\Rightarrow {}^*W$ რომელიმე $W\in \sum^*$ ჯაჭვისათვის

ბაზისი: i=0, არ საჭიროებს დამტკიცებას, რადგან $N_0\neq \varnothing$.

ვიგულისხმოთ, რომ (2.4.1) ჭეშმარიტია i-სთვის, და ავიღოთ $A\in N_{i+1}$. თუ A ეკუთვნის აგრეთვე N_i_ს, მაშინ ინდუქციის ნაბიჯი ტრივიალურია. თუ $A\in N_{i+1}-N_i$, მაშინ არსებობს წესი $A\rightarrow X_1X_2...X_k$, სადაც ყოველი X_i სიმბოლო ეკუთვნის ან $\sum$ ან N_i-ს. ამგვარად, ყოველი X_j-სთვის შეიძლება ვიპოვოთ ისეთი W_j ჯაჭვი, რომ $X_j\Rightarrow {}^*W$ თუ $X_j\in \sum$, მაშინ $W=X_j$, წინააღმდეგ შემთხვევაში W_j -ს არსებობა გამომდინარეობს (2.4.1)- დან. ადვილად ჩანს, რომ

$A\Rightarrow X_1...X_k\Rightarrow {}^*W_1X_2...X_k\Rightarrow {}^*...\Rightarrow {}^*W_1W_2...W_k$

შემთხვევა K=0 (ე.ი. $A\rightarrow e$ წესი) არაა გამორიცხული. ინდუქციის ნაბიჯი დამთავრებულია.

N_i სიმრავლეების განმარტება უზრუნველყოფს, რომ თუ $N_i=N_{i+1}$, მაშინ $N_i=N_{i+1}=...$ ჩვენ უნდა ვაჩვენოთ, რომ თუ $A\Rightarrow {}^*W$ რომელიმე $W\in \sum^*$ ჯაჭვისათვის, მაშინ $A\in N_e$. ზემოთ გაკეთებული შენიშვნის ძალით ყველაფერი, რის ჩვენებაც საჭიროა ესაა ის, რომ $A\in N_i$ რომელიმე i-სთვის. n-ის მიხედვით ინდუქციით დავამტკიცოთ, რომ

თუ, $A\Rightarrow {}^nW$, მაშინ $A\in N_i$ რომელიმე i-სთვის. ბაზისი:n=1, ტრივიალურია, რადგან ამ შემთხვევაში i=1. დავუშვათ,რომ ჭეშმარიტია n-სთვის,და ვთქვათ $A\Rightarrow {}^{n+1}W$. მაშინ შეიძლება დაიწეროს $A\Rightarrow X_1...X_k\Rightarrow {}^nW$, სადაც $W=W_1...W_k$ ჯაჭვი ისეთია, რომ $X_i\Rightarrow {}^{nj}W_j$ ყოველი j-სა და $n_j\leq$ n-სთვის.

-ის თანახმად, თუ $X_j \in N$, მაშინ $X_j \in N_i$ რომელიმე j-სთვის. თუ $X_j \in \Sigma$, მაშინ $i_j=0$. დავუშვათ, $i=1+\max(i_1,...,i_k)$, მაშინ განმარტების თანახმად $A \in N_i$ და ინდუქცია დამთავრებულია.

ჩავსვათ A=S (-სა და -ში და მივიღებთ თეორემის დამტკიცებას.□

შედეგი. G კონტექსტისაგან თავისუფალი გრამატიკისათვის L(G) ენის სიცარიელის პრობლემა გადაწყვეტადია. □

განმარტება 2.3.2. $X \in (N \cup \Sigma)$ სიმბოლოს ვუწოდოთ მიუღწევადი CF გრამატიკაში $G=(N,\Sigma,P,S)$, თუ X არ გვხვდება არცერთ გამოყვანად ჯაჭვში.

მიუღწევადი სიმბოლოები შეიძლება მოვიცილოთ CF გრამატიკიდან შემდეგი ალგორითმის საშუალებით:

ალგორითმი 2.3.2. მიუღწევადი სიმბოლოების მოცილება.

შესასვლელი. CF გრამატიკა $G=(N,\Sigma,P,S)$.

გამოსასვლელი. CF გრამატიკა $G' =(N,' \Sigma',P',S)$ რომლისთვისაც

1. $L(G')=L(G)$,

2. ყოველი $X \in N' \cup \Sigma'$ - არსებობს ისეთი α და β ჯაჭვები $(N' \cup \Sigma')$-დან, რომ

$S \Rightarrow_{G1}^{*} \alpha X \beta$.

მეთოდი.

1. დავუშვათ $V_0=\{ S\}$ და i=1.

2. დავუშვათ $V_i=\{ X|$ P-ში არის $A \to \alpha X \beta$ და $A \in V_{i-1}\} \cup V_{i-1}$.

3. თუ $V_i \neq V_{i-1}$, მაშინ დავუშვათ i=i+1 და გადავიდეთ (2) ნაბიჯზე. წინააღმდეგ შემთხვევაში, ვთქვათ

$N' =V_i \cap N$,

$\Sigma' =V_i \cap \Sigma$

P' შედგება P სიმრავლის ისეთი წესებისაგან, რომლებიც შეიცავენ მხოლოდ სიმბოლოებს V_i-საგან, $G' =(N',\Sigma',P',S)$.□

და ალგორითმები ძალიან გვანან ერთიმეორეს.შევნიშნოთ რომ ალგორითმის მე-2 ნაბიჯი შეიძლება გავიმეოროთ მხოლოდ სასრულო რიცხვჯერ, რადგანაც $V_i \leq N \cup \Sigma$. გარდა ამისა, i-ს მიხედვით ინდუქციით პირდაპირი დამტკიცება გვიჩვენებს, რომ $S \Rightarrow G' \alpha X\beta$, მაშინ და მხოლოდ მაშინ,როცა $X \in V_i$ რომელიმე i-სთვის. ახლა ჩვენ შეგვიძლია მოვიცილოთ CF გრამატიკიდან ყველა უსარგებლო სიმბოლო.

ალგორითმი2.3.3. უსარგებლო სიმბოლოების მოცილება.

შესასვლელი. CF გრამატიკა $G=(N,\Sigma,P,S)$, რომელიც $L(G) \neq \varnothing$.

გამოსასვლელი. CF გრამატიკა $G' =(N' ,\Sigma' ,P' ,S)$, რომლისთვისაც $L(G')=L(G)$ და $(N' \cup \Sigma')$-ში არაა უსარგებლო სიმბოლოები.

მეთოდი.

1. G-ზე ალგორითმის გამოყენებით მივიღოთ N_e.დავუშვათ $G_1=(N \cap N_e,\Sigma ,P_1,S)$, სადაც P_1 შედგება P სიმრავლის ისეთი წესებისაგან,რომლებიც შეიცავენ მხოლოდ $N_e \cup \Sigma$ -ს სიმბოლოებს.

2. G-ზე ალგორითმის გამოყენებით მივიღოთ $G' =(N' ,\Sigma' ,P' ,S)$.□

ალგორითმის 1 ნაბიჯზე G-ს ჩამოსცილდება ყველა ისეთი არატერმინალები, რომლებიც არ წარმოშობენ ტერმინალურ ჯაჭვებს. შემდეგ 2 ნაბიჯზე ჩამოსცილდებიან ყველა მიუღწევადი სიმბოლოები. ყოველი X სიმბოლო მიღებული გრამატიკიდან უნდა მონაწილეობდეს ერთ მაინც $S \Rightarrow {}^{*}WXY \Rightarrow {}^{*}WXY$ სახის გამოყვანაში. შევნიშნოთ, რომ თუ თავდაპირველად გამოვიყენებთ ალგორითმს და შემდეგ - ალგორითმს, მაშინ ყოველთვის ვერ მივიღებთ გრამატიკას,რომელიც არ შეიცავს უსარგებლო სიმბოლოებს.

თეორემა 2.3.3. G' გრამატიკა,რომელსაც აგებს ალგორითმი არ შეიცავს უსარგებლო სიმბოლოებს და

$L(G)=L(G')$.

დამტკიცება. იმის დამტკიცებას, რომ $L(G)=L(G')$ მარტივია. ვიგულისხმოთ, რომ $A \in N'$ უსარგებლო სიმბოლოა. მაშინ უსარგებლო სიმბოლოს განმარტების თანახმად შეიძლება წარმოგვიდგეს ორი შემთხვევა:

1 შემთხვევა: $S \Rightarrow_{G1}^{*\alpha} A\beta$ გამოყვანა შეუძლებელია როგორიც არ უნდა იყოს α და β . ასეთ შემთხვევაში A სიმბოლოს ჩამოცილება მოხდება ალგორითმის მე-2 ნაბიჯის დროს.

2 შემთხვევა: $S \Rightarrow_{G1}^{*} \alpha A\beta$ რომელიმე α და β -სთვის, მაგრამ $A \Rightarrow_{G1}^{*} W$ გამოყვანა $W \in \sum'^{*}$ არ არსებობს. მაშინ A არ ჩამოსცილდება მე-(2) ნაბიჯზე და, გარდა ამისა, თუ $A \Rightarrow_{G}^{*} \gamma B\delta$, მაშინ B-ც არ ჩამოსცილდება მე-(2) ნაბიჯზე. ამგვარად, თუ $A \Rightarrow_{G}^{*} W$, მაშინ $A \Rightarrow_{G1}^{*} W$.

აქედან, შეიძლება დავასკვნათ, რომ $A \Rightarrow_{G}^{*} W$ $W \in \sum^{*}$-სთვის არ არსებობს და A ჩამოსცილდება (1) ნაბიჯზე.

იმის დამტკიცება, რომ G′ -ის არცერთი ტერმინალი არ შეიძლება იყოს უსარგებლო ჩატარდება ანალოგიურად.□

მაგალითი 2.3.1. განვიხილოთ G=({ S,A,B} ,{ a,b} ,P,S) გრამატიკა, სადაც P შედგება წესებისაგან:

$S \rightarrow a \mid A$

$A \rightarrow AB$

$B \rightarrow b$

გამოვიყენოთ G-ს მიმართ ალგორითმი. (1) ნაბიჯზე მივიღებთ N_e={ S,B} დაG_1=({ S,B} ,{ a,b} ,{ $S \rightarrow a, B \rightarrow b$} ,S).

ალგორითმის გამოყენებით,მივიღებთ $V_2=V_1$={ S,a} . ამგვარად. G′ =({ S} ,{ a} ,{ $S \rightarrow$ a} ,S).

თუ G-ს მიმართ გამოვიყენებთ ჯერ ალგორითმს ,მაშინ აღმოჩნდება,რომ ყველა სიმბოლო მიუცწევადია,ასე რომ გრამატიკა არ შეიცვლება. შემდეგ ალგორითმის გამოყენება მოგვცემს N_e={S,B} და შედეგად გვექნება გრამატიკა G_1, რომელიც განსხვავდება G′ -საგან.□

ხშირად ხელსაყრელია CF გრამატიკიდან G მოვიცილოთ e წესები, ე.ი. $A \rightarrow e$ სახის წესები. მაგრამ თუ $e \in L(G)$, მაშინ ცხადია, რომ $A \rightarrow e$ სახის წესებს ვერ მოვიცილებთ.

განმარტება 2.3.4. ვუწოდოთ CF გრამატიკას G=(N,$\sum$,P,S) გრამატიკა e წესების გარეშე (ან არადაპატარავებადი), თუ

1. P არ შეიცავს e წესებს, ან

2. არსებობს ზუსტად ერთი e წესი $S \to e$ და S არ გვხვდება დანარჩენი წესების მარჯვენა მხარეში.

ალგორითმი 2.3.5. e წესების გარეშე გრამატიკად გარდაქმნა.

შესასვლელი. CF გრამატიკა $G=(N,\Sigma,P,S)$.

გამოსასვლელი. ექვივალენტური CF გრამატიკა $G' =(N' ,\Sigma ,P' ,S')$ e წესების გარეშე.

მეთოდი.

(1) ავაგოთ $N_e=\{ A|\ A\in N$ და $A\Rightarrow {}^{+}{}_{G}e\}$ ეს ანალოგიურია იმისა, რაც იყო 1 და 2 ალგორითმებში და ვტოვებთ დამტკიცების გარეშე.

(2) ავაგოთ P′ ისე, რომ

a. თუ $A\to \alpha_0 B_1 \alpha_1 B_2 \alpha_2 ... B_k \alpha_k$ ეკუთვნის $P, K\geq O$ და $B_i\in N_e$ $1\leq i\leq k$, ხოლო არცერთი α_j ჯაჭვის სიმბოლო $(O\leq J\leq K)$ არ ეკუთვნის N_e-ს, მაშინ P^i-ში ჩავრთოთ ყველა წესები $A\to \alpha_0 X_1 \alpha_1 X_2 ... \alpha_{k-1} X_k \alpha_k$, სადაც X_i არის ან B_i ან e, მაგრამ არ ჩავრთოთ $A\to e$ წესი (ეს შეიძლება იმ შემთხვევაში, თუ ყველა α_i ტოლია e-სი).

b. თუ $S\in Ne$, მაშინ P′ -ში ჩავრთოთ წესი

$S' \to e|\ S,$

სადაც S′ ახალი სიმბოლოა. დავუშვათ, $N' =N\cup \{ S' \}$.

წინააღმდეგ შემთხვევაში დავუშვათ, რომ $N' =N$ და $S' =S$.

(3) მივიღოთ , რომ $G' =(N' ,\Sigma ,P' ,S')$.□

2.23 მაგალითი. განვიხილოთ გრამატიკა 2.19 მაგალითიდან წესებით

$S\to aSbS|\ bSaS|\ e$ ამ გრამატიკისათვის 2.10 ალგორითმის გამოყენებით, მივიღებთ გრამატიკას

$S' \to S|\ e$

$S\to aSbS|\ bSaS|\ aSb|\ abS|\ ab|\ bSa|\ baS|\ ba$□

თეორემა 2.3.5. ალგორითმი გვაძლევს გრამატიკას e წესების გარეშე, რომელიც შესავალი გრამატიკის ექვივალენტურია.

დამტკიცება. უშუალოდ ჩანს,რომ ალგორითმი გვაძლევს G′ გრამატიკას e წესების გარეშე. იმის საჩვენებლად, რომ L(G)=L(G′), საკმარისია დავამტკიცოთ ინდუქციით W ჯაჭვის სიგრძის მიხედვით, რომ $A\Rightarrow \bullet\ {}_{G1}W$ მაშინ და მხოლოდ მაშინ,როცა W≠ e და $A\Rightarrow \bullet\ {}_{G}W$. რის დამტკიცებაც მარტივია. ჩავსვათ S A-ს ნაცვლად -ში. ვხედავთ, რომ W∈ L(G) W≠ e-სთვის მხოლოდ და მხოლოდ მაშინ,როცა W∈ L(G′). ცხადია, რომ e∈ L(G) მხოლოდ და მხოლოდ მაშინ, როცა e∈ L(G′). ამგვარად, L(G)=L(G′). □

გრამატიკის სხვა სასარგებლო გარდაქმნაა $A\rightarrow B$ სახის წესების მოცილება, რომლებსაც ჩვენ ვუწოდებთ ჯაჭვურ წესებს.

ალგორითმი 2.3.6. ჯაჭვური წესების მოცილება.

შესასვლელი. CF გრამატიკა G e წესების გარეშე.

გამოსასვლელი. ექვივალენტური CF გრამატიკა G′ e წესებისა CF

ჯაჭვური წესების გარეშე.

მეთოდი.

1. ყოველი A∈ N-სთვის ავაგოთ $N_A=\{ B|\ A\Rightarrow \bullet B\}$

შემდეგნაირად:

a. მივიღოთ $N_0=\{ A\}$ და i=1.

b. მივიღოთ $N_i=\{ C|\ B\rightarrow c$ ეკუთვნის P და $B\in N_{i-1}\} \cup N_{i-1}$.

c. თუ $N_i \neq N_{i-1}$ დავუშვათ i=i+1 და (b) ნაბიჯი გავიმეოროთ. წინააღმდეგ შემთხვევაში მივიღოთ $N_A=N_i$.

2. ავაგოთ P′ ასე: თუ $B\rightarrow \alpha$ ეკუთვნის P-ს და იგი ჯაჭვური წესი არაა, ჩავრთოთ P′ -ში $A\rightarrow \alpha$ წესი ყველა ისეთი A-სთვის, რომ $B\in N_A$.

3. მივიღოთ G′ =(N,∑ ,P′ ,S). □

მაგალითი. გამოვიყენოთ 2.11 ალგორითმი G_0 გრამატიკისათვის, რომლის წესებია:

$E\rightarrow E+T|\ T$

$T\rightarrow T*F|\ F$

$F\rightarrow (E)|\ a$

(1) ნაბიჯზე N_E={ E,T,F}, N_T={ T,F}, N_F={ F}. (2) ნაბიჯის შემდეგ P′ სიმრავლე იქნება ასეთი:

E→ E+T| T*F| (E)| a

T→ T*F| (E)| a

F→ (E)| a. □

თეორემა 2.3.7. G′ გრამატიკას, რომელსაც აგებს ალგორითმი გვაძლევს G′ გრამატიკას, რომელსაც ჯაჭვური წესები არ გააჩნია.

თავდაპირველად ვაჩვენოთ, რომ L(G′)≤L(G). ვთქვათ, W∈L(G′). მაშინ, G′ გრამატიკაში არსებობს გამოყვანა $S\Rightarrow \alpha_0 \Rightarrow \alpha_1\Rightarrow .\Rightarrow \alpha_n^W$ თუ α_i-დან α_{i+1}-ზე გადასვლისას A→ β წესი გამოიყენება, მაშინ არსებობს ისეთი სიმბოლო B∈ N შესაძლებელია B=(A), რომ $A\Rightarrow_G B$

და $B\Rightarrow_G \beta$. ამგვარად, $A\Rightarrow_G \beta$ და $\alpha_i\Rightarrow_G \alpha_{i+1}$ აქედან გამომდინარეობს, რომ $S\Rightarrow_G W$ და W∈ L(G), ისე რომ L(G′)≤ L(G).

ახლა ვაჩვენოთ, რომ L(G)≤ L(G′). ვთქვათ W∈ L(G) და $S\Rightarrow \alpha_{0\Rightarrow}\alpha_{1\Rightarrow}...\Rightarrow\alpha_n$=W- W ჯაჭვის მარცხენა გამოყვანაა G გრამატიკაში. შეიძლება ვიპოვოთ $i_1,i_2,...i_k$ ინდექსების მიმდევრობა, რომელიც შედგება ზუსტად იმ J-სგან, რომელთათვისაც ნაბიჯზე $\alpha_{j-1\Rightarrow}\alpha_j$ გამოიყვანება არა ჯაჭვური წესით.

რადგანაც ჩვენ ვიხილავთ მარცხენა გამოყვანას, ამიტომ ჯაჭვური წესების მიმდევრობით გამოყენება ცვლიან სიმბოლოს, რომელსაც უჭირავს ერთი და იგივე პოზიცია მარცხენა გამოყვანად ჯაჭვებში. აქედან სჩანს, რომ $S\Rightarrow_{G1}\alpha_{i1\Rightarrow}\alpha_{i2\Rightarrow G1}...\Rightarrow_{G1}\alpha_{ik}$=W. ამგვარად, W∈ L(G′), რაც იმას ნიშნავს, რომ L(G′)=L(G). □

განმარტება 2.3.7. CF გრამატიკას G=(N,∑,P,S) ეწოდება გრამატიკა ციკლების გარეშე,თუ მასში არაა გამოყვანა $A\Rightarrow^+A$ A∈ N. G გრამატიკას ეწოდება დაყვანილი, თუ იგი არ შეიცავს ციკლებს, e-წესებს და უსარგებლო სიმბოლოებს.

e-წესების ან ციკლების მქონე გრამატიკების ანალიზები ზოგჯერ უფრო ძნელია, ვიდრე e-წესების არ მქონე და უციკლო გრამატიკების ანალიზი. გარდა ამისა, ნებისმიერ პრაქტიკულ სიტუაციაში უსარგებლო სიმბოლოები აუცილებლობის გარეშე ზრდიან ანალიზატორის მოცულობას. ამიტომ სინტაქსური ანალიზის ზოგიერთი ალგორითმებისათვის, ჩვენ მოვითხოვთ, რომ მათში ფიგურირებული გრამატიკები იყვნენ დაყვანილნი. დავამტკიცოთ, რომ ამ მოთხოვნის მიუხედავად მაინც განიხილება ნებისმიერი CF ენები.

თეორემა 2.3.8. თუ L-CF ენაა,მაშინ L=L(G) რომელიმე დაყვანილი CF გრამატიკისათვის G.

დამტკიცება. გამოვიყენოთ ალგორითმები CF გრამატიკისათვის, რომელიც განსაზღვრავს L ენას.□

განმარტება 2.3.8.CF გრამატიკის A წესი ეწოდება A→ α სახის წესს (არ აგერიოთ A წესი e-წესში, რომელსაც აქვს სახე B→ e.

შემოვიტანოთ კიდევ გარდაქმნა, რომლის საშუალებითაც შეიძლება ამოვაგდოთ გრამატიკიდან ერთი A→ α Bβ სახის წესი. იმისათვის, რომ ამოვაგდოთ ეს წესი, საჭიროა გრამატიკას დავუმატოთ ახალი წესები, რომლებიც მიიღება მასში B არატერმინალის შეცვლით მარჯვენა მხარით ყველა B წესიდან.

ლემა. ვთქვათ G=(N,∑ ,P,S)-CF გრამატიკაა და P შეიცავს წესს A→ α Bβ , სადაც B∈ N, ხოლო α და β ეკუთვნიან $(N\cup \sum)^*$. ვთქვათ B→ $\gamma_1|\gamma_2|...|\gamma_k$ ამ გრამატიკის ყველა B წესია. ვთქვათ

G′ =(N,∑ ,P′ ,S), სადაც

P′ =(P-{ A→ α Bβ })∪ { A→ $\alpha\gamma_1\beta | \alpha\gamma_2\beta | ... | \alpha\gamma_k\beta$ }

მაშინ L(G)=L(G′). დამტკიცება მარტივია. □

მაგალითი 2.3.5. მოვიცილოთ A→ aAA წესი G გრამატიკიდან, რომლის ყველა A წესია A→ aAA| b. 2.14 ლემის გამოყენებით ვიგულისხმოთ α=a, B=A და β =A და მივიღოთ G′ გრამატიკა წესებით A→ aaAAA| abA| b

aabbb ჯაჭვისათვის გამოყვანის ხეები G და G′ გრამატიკებში ნაჩვენებია 2.12 ნახატზე. შევნიშნოთ, ამ გარდაქმნის ეფექტი მდგომარეობს a ნახატზე მოცემული ხის ძირის მიწებებაში მის მეორე პირდაპირ შთამომავალთან.□

A A

a A A a a A A A

a A A b b b b

b b

a b

ნახ. გამოყვანის ხეები: a– G გრამატიკაში; b– G′ გრამატიკაში. აკლია ნახაზი

2.4 ხომსკის ნორმალური ფორმა

განმარტება 2.4.1 CF გრამატიკას G=(N,∑,P,S) ეწოდება გრამატიკა ხომსკის ნორმალურ ფორმაში (ანდა ბინარულ ნორმალურ ფორმაში), თუ ყოველ წესს P– დან აქვს ერთ-ერთი შემდეგი სახეებიდან:

1. $A \rightarrow BC$, სადაც A,B და C ეკუთვნიან N-ს,

2. $A \rightarrow a$, სადაც $a \in \sum$,

3. S–e, თუ $e \in L(G)$ და S არ გვხვდება სხვა წესების მარჯვენა მხარეში. ვაჩვენოთ,რომ ყოველი CF ენა მიიღება ხომსკის ნორმალურფორმიანი გრამატიკიდან. ეს შედეგი სასარგებლოა იმ შემთხვევებში, როცა გვჭირდება CF ენის წარმოდგენის მარტივი ფორმა.

ალგორითმი 2.4.1 ხომსკის ნორმალურ ფორმად გარდაქმნა.

შესასვლელი. დაყვანილი CF გრამატიკა G=(N,∑ ,P,S).

გამოსასვლელი. CF გრამატიკა G′ ხომსკის ნორმალურ ფორმაში, რომელიც G-ს ექვივალენტურია, ე.ი. L(G′)=L(G).

მეთოდი. გრამატიკა G′ იგება G-სგან შემდეგნაირად:

1. ჩავრთოთ P′ -ში $A \rightarrow a$ სახის ყოველი წესი P-დან.

2. ჩავრთოთ P′ -ში $A \rightarrow BC$ სახის ყოველი წესი P-დან.

3. ჩავრთოთ P′ -ში $A \rightarrow e$ წესი, თუ იგი იყო P-ში.

4. $A \rightarrow X_i...X_k$ სახის ყოველი წესისათვის P-დან, სადაც K> 2 ჩავრთოთ P′ -ში წესები

$A \rightarrow X'_1 < X_2...X_k >$

$< X_2...X_k > \rightarrow X'_2 < X_3...X_k >$

.

.

.

$< X_{k-2} X_{k-1}X_k> \rightarrow X'_{k-2}< X_{k-1}X_k>$

$< X_{k-1}X_k> \rightarrow X'_{k-1}X'_k$

სადაც $X'_i=X_i$, თუ $X_i \in N$; X'_i – ახალი არატერმინალია, თუ $X_i \in \Sigma$;

$< X_i...X_k>$ – ახალი არატერმინალია.

5. $A \rightarrow X_1X_2$ სახის ყოველი წესისათვის P-დან, სადაც ერთი მაინც X_1X_2 სიმბოლოებიდან ეკუთვნის Σ , ჩავრთოთ P′ -ში $A \rightarrow X'_1X'_2$ წესი.

6. a′ სახის ყოველი არატერმინალისთვის, რომლებიც შემოტანილია (4) და (5) ნაბიჯებზე, ჩავრთოთ P′-ში $a' \rightarrow a$ წესი. და ბოლოს, N′ შეიცავს N-ს და ყველა ახალ არატერმინალებს, რომლებიც შემოტანილია P′ აგების დროს. მაშინ საძიებელი გრამატიკა იქნება G′ =(N′ ,Σ ,P′ ,S).□

თეორემა 2.4.2. ვთქვათ L არის CF ენა. მაშინ L=L(G′) რომელიმე CF გრამატიკისათვის G′ ხომსკის ნორმალურ ფორმაში.

დამტკიცება. თეორემის თანახმად L განისაზღვრება დაყვანილი G გრამატიკით. ალგორითმი G-დან აგებს G′ გრამატიკას, რომელიც ცხადია, წარმოდგენილია ხომსკის ნორმალური ფორმით. დაგვრჩენია ვაჩვენოთ, რომ L(G)=L(G′). ეს მტკიცდება ლემის გამოყენებით G′ გრამატიკის ყოველი წესისათვის, რომელთა მარჯვენა მხარეში შედის a′, და შემდეგ წესებისათვის $<X_i...X_j>$ სახის არატერმინალებით. □

მაგალითი 2.4.1. ვთქვათ G დაყვანილი გრამატიკაა, განსაზღვრული წესებით:

$S \rightarrow aAB| BA$

$A \rightarrow BBB| a$

$B \rightarrow AS| b$

ვაგებთ P′ 2.12 ალგორითმით, ვტოვებთ რა წესებს $S \rightarrow BA$, $A \rightarrow a$, $B \rightarrow AS$ და $B \rightarrow b$. ვცვლით $S \rightarrow aAB$ წესებით:

$S \rightarrow a' < AB>$ და $< AB> \rightarrow AB$, ხოლო $A \rightarrow BBB$-ს წესებით – $A \rightarrow B< BB>$ და $< BB> \rightarrow BB$. და ბოლოს,ვუმატებთ $a' \rightarrow a$, შედეგად ვიღებთ გრამატიკას G′ =(N′ ,{ a,b} ,P′ ,S), სადაც

N′ ={ S,A,B,< AB> ,< BB> ,a′ } , ხოლო P′ შედგება წესებისაგან

S→ a′ < AB> | BA

A→ B< BB> | a

B→ AS| b

< AB> → AB

< BB> → BB

a′ → a □

2.5 გრეიბახის ნორმალური ფორმა

ახლა ვაჩვენოთ, რომ ყოველი CF ენისათვის შეიძლება ვიპოვოთ გრამატიკა, რომელშიც წესების მარჯვენა მხარეები იწყებიან ტერმინალებით. ასეთი გრამატიკის აგება დაფუძნებულია ეგრეთწოდებული მარცხენა რეკურსიის აცილებაზე.

განმარტება 2.5.1. G=(N,∑ ,P,S) CF გრამატიკის A არატერმინალს ეწოდება რეკურსიული, თუ A⇒ $^{+\alpha}$ Aβ რომელიმე α და β -სთვის. თუ α =e, მაშინ A-ს ეწოდება მარცხნივ რეკურსიული. ანალოგიურად, თუ β =e, მაშინ A-ს ეწოდება მარჯვნივ რეკურსიული. გრამატიკას, რომელსაც გააჩნია ერთი მაინც მარცხნივ რეკურსიული არატერმინალი, ეწოდება მარცხნივ რეკურსიული.

ანალოგიურად, განიმარტება მარჯვნივ რეკურსიული გრამატიკა. გრამატიკას, რომელშიც ყველა არატერმინალები, გარდა შეიძლება საწყისი სიმბოლოსა, რეკურსიულებია ეწოდება რეკურსიული.

ზოგიერთ გარჩევის ალგორითმებს, რომლებზეც ჩვენ შემდგომში განვიხილავთ არ შეუძლიათ იმუშაონ მარცხნივ რეკურსიულ გრამატიკებზე. ვაჩვენოთ, რომ ნებისმიერი CF ენა განისაზღვრება ერთი მაინც არა მარცხნივ რეკურსიული გრამატიკით. დავიწყოთ CF გრამატიკიდან უშუალოდ მარცხნივ რეკურსიულობის ჩამოშორებით.

ლემა 2.5.1. ვთქვათ G=(N,∑ ,P,S)-CF გრამატიკაა, რომელშიც

A→ Aα $_1$| Aα $_2$| ...| Aα $_m$| β $_1$| β $_2$| ...| β $_n$

ყველა A წესებია P-დან და არცერთი β_i (i=1,2,...n) ჯაჭვებიდან არ იწყება A-თი. ვთქვათ,

G′ =(N∪ { A′ } ,∑ ,P′ ,S),

სადაც A′ -ახალი არატერმინალია, ხოლო P′ მიიღება P-საგან A წესების შეცვლით

$A \rightarrow \beta_1 | \beta_2 | ... | \beta_n | \beta_1 A' | \beta_2 A' | ... | \beta_n A'$

$A' \rightarrow \alpha_1 | \alpha_2 | ... | \alpha_m | \alpha_1 A' | \alpha_2 A' | ... | \alpha_m A'$

წესებით: მაშინ L(G′)=L(G).

დამტკიცება. ჯაჭვები, რომლებიც შეიძლება მივიღოთ G გრამატიკაში A არატერმინალისაგან A წესების გამოყენებით ყველაზე მარცხენა არატერმინალზე ჰქმნიან $(\beta_1+\beta_2+...+\beta_n)(\alpha_1+\alpha_2+...+\alpha_m)^*$ რეგულარულ სიმრავლეს. ეს ზუსტად ის ჯაჭვებია, რომლებიც შეიძლება მივიღოთ G′ -ში A-სგან მარჯვენა გამოყვანებით, გამოვიყენებთ რა ერთხელ A წესს და რამოდენიმეჯერ A′ წესებს (შედეგად მთელი გამოყვანა არ იქნება მარცხენა). გამოყვანის ყველა ნაბიჯები G-ში,რომლებშიც არ გამოიყენება A წესები შეიძლება უშუალოდ ჩავატაროთ G′ -ში, რადგანაც A წესების გარდა G და G′ ერთნაირია. აქედან შეიძლება დავასკვნათ, რომ L(G′)≤ L(G).

შებრუნებული ჩართვა მტკიცდება არსებითად ისევე. G′-ში აიღება მარჯვენა გამოყვანა და განიხილებიან ნაბიჯების მიმდევრობები, რომლებიც შედგებიან მხოლოდ ერთხელ A წესების გამოყენებისაგან და რამოდენიმეჯერ A′ წესების გამოყენებებისაგან. ამგვარად, L(G)=L(G′). □

ნახაზზე 2.5.1 ნაჩვენებია, როგორ მოქმედებს გამოყვანის ხეებზე ლემაში აღწერილი გარდაქმნა.

A A

A α_{i1} ß A'

A α_{i2} α_{ik} A'

.

. α_{ik-1} A'

.

A .

.

A α_{ik} .

A'

β

α_{i2} A'

α_{i1}

a b

α_{ik}

A

β

a b

ნახ. ხის ნაწილები: a-ხის ნაწილი G-ში. b-ს შესაბამისი ნაწილი G′ -ში.

მაგალითი 2.5.2. ვთქვათ, G_0 ჩვენი ჩვეულებრივი გრამატიკაა წესებით.

E→ E+T| T

T→ T*F| F

F→ (E)| a

თუ ამ გრამატიკის მიმართ გამოვიყენებთ ლემის კონსტრუქციას, მაშინ მიიღება ექვივალენტური G′ გრამატიკა წესებით

E→ T| TE′

E′ → +T| +TE′

T→ F| FT′

$T' \rightarrow F|\ FT'$

$F \rightarrow (E)|\ a.$ □

ახლა, ჩვენ მზად ვართ აღვწეროთ ალგორითმი, რომელიც დაყვანილი CF გრამატიკას მოაშორებს მარცხენა რეკურსიას. თავისი იდეით ეს ალგორითმი ჰგავს რეგულარულ კოეფიციენტებიანი განტოლების ამოხსნის ალგორითმს.

2.6 მარცხენა რეკურსიის მოცილება

ალგორითმი 2.6.1. მარცხენა რეკურსიის მოცილება.

შესასვლელი. დაყვანილი CL გრამატიკა G=(N,Σ ,P,S).

გამოსასვლელი. ექვივალენტური CL გრამატიკა G′ მარცხენა რეკურსიის გარეშე.

მეთოდი.

1. ვთქვათ, $N=\{ A_1,...,A_n\}$. გარდავქმნათ G ისე, რომ $A_i \rightarrow \alpha$ წესში α ჯაჭვი იწყებოდეს ტერმინალით ან ისეთი A_j-თი, რომ j> i. ამ მიზნით ვთქვათ i=1.

2. ვთქვათ, A_i წესების სიმრავლეა $A_i \rightarrow A_i\alpha_1| ...| A_i\alpha_m| \beta_1| ...\beta_p$, სადაც არცერთი β_j ჯაჭვებიდან არ იწყება A_k-თი, თუ K≤ i. (ეს ყოველთვის შესაძლებელია). შევცვალოთ A_i წესები

$A_i \rightarrow \beta_i| ...| \beta_p| \beta_1 A'_i| ...| \beta_p A'_i$

$A'_i \rightarrow \alpha_1| ...| \alpha_m| \alpha_1 A'_i| ...| \alpha_m A'_i$

წესებით, სადაც A'_i ახალი არატერმინალია. ყველა A_i წესების მარჯვენა მხარეები ახლა იწყებიან ტერმინალით ანდა A_k-თი, სადაც K> i.

3. თუ i=n, მაშინ მიღებული გრამატიკა ჩავთვალოთ საბოლოო შედეგად და გავჩერდეთ. წინააღმდეგ შემთხვევაში მივიღოთ i+1 და j=1.

4. შევცვალოთ ყოველი $A_i \rightarrow Aj\alpha$ სახის წესი წესებით $A_i \rightarrow \beta_1\alpha | ...| \beta_m\alpha$,

სადაც $A_j \rightarrow \beta_1| \beta_2| ...| \beta_m$ ყველა A_j წესებია. რადგანაც ყოველი A_j წესის მარჯვენა მხარე იწყება უკვე ტერმინალით ან A_k-თი K>j-სთვის, ამიტომ ყოველი A_i წესის მარჯვენა მხარესაც, ახლა ექნება ეს თვისება.

5. თუ j=i-1 გადავიდეთ მე-(2) ნაბიჯზე. წინააღმდეგ შემთხვევაში მივიღოთ j=j+1 და გადავიდეთ მე-(4) ნაბიჯზე.□

თეორემა 2.6.1. ყოველი CF ენა განისაზღვრება არამარცხნივრეკურსიული გრამატიკით.

დამტკიცება. ვთქვათ G დაყვანილი გრამატიკაა, რომელიც წარმოშობს KL L ენას. მასზე ალგორითმის გამოყენებისას, გამოიყენება მხოლოდ ის გარდაქმნები, რომლებიც მოხსენებულია და ლემებში. ამიტომ საბოლოო G′ გრამატიკა წარმოქმნის L ენას.

ჩამოვაყალიბოთ ორი დებულება, რომლებიც არსებითად უკვე შეგვხვდა ალგორითმის მე-(2) და მე-(4) ნაბიჯების აღწერების ბოლოში:

i-სთვის მე-(2) ნაბიჯის შესრულების შემდეგ ყოველი A_i წესის მარჯვენა მხარე იწყება ტერმინალით ან ისეთი A_k-თი, რომ K> I.

ავტომატები მაღაზიური მეხსიერებით

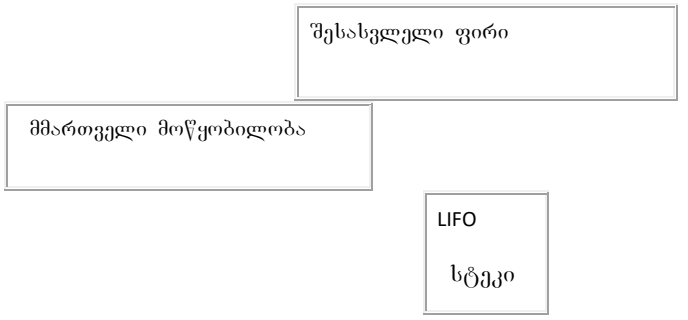
დაპროგრამების ცნობილი ენები განისაზღვრებიან კონტექსტისაგან თავისუფალი გრამატიკებით. ეს იმას ნიშნავს, რომ თუ ჩვენ ავიღებთ რაიმე დაპროგრამების ენას მაგალითად, FORTRAN-IV-ს ჩვენ შეგვიძლია შევადგინოთ კონტექსტისაგან თავისუფალი გრამატიკა GF, რომელიც უშვებს მხოლოდ FORTRAN-IV-ზე სინტაქსურად სწორად ჩაწერილ პროგრამებს. ე. ი. ნებისმიერი FORTRAN-IV-ზე სწორად ჩაწერილი პროგრამისათვის, რომელიც არის რაიმე P ტექსტი, მოიძებნება GF გრამატიკაში გამოყვანა $S\Rightarrow^*P$ და რაიმე P_1 არასწორად ჩაწერილი პროგრამისათვის არ არსებობს $S\Rightarrow^*P_1$ GF გრამატიკაში. აქედან გამომდინარეობს, რომ თუ ჩვენ გვექნება ალგორითმი $S\Rightarrow^*P$ ნებისმიერი P-ს საპოვნელად, მაშინ ჩვენ შეგვიძლია ამ ალგორითმის რეალიზაცია კომპიუტერზე SA პროგრამის სახით და მისი საშუალებით შევამოწმოთ P პროგრამა არის თუ არა FORTRAN-IV-ზე სწორად ჩაწერილი პროგრამა. ასეთ SA პროგრამას ეწოდება სინტაქსური ანალიზატორი. იგივე ამოცანა შეიძლება გადაწყვეტილ იქნეს ნებისმიერი კონტექსტისაგან თავისუფალი გრამატიკისათვის სპეციალური მოწყობილობით, რომელსაც ეწოდება ავტომატი მაღაზიური მეხსიერებით. ასეთი ავტომატები სასრული ავტომატებისაგან ძირითადად განსხვავდებიან იმით, რომ მათ შეუძლიათ სპეციალურ მეხსიერებაში დაიმახსოვრონ თუ რა მდგომარეობები გაიარეს მოცემული მომენტისათვის და ამით აღადგინოთ წინა

ისტორია, რომელიც გაიარა ავტომატმა საწყისი მდგომარეობიდან მოცემულ მდგომარეობამდე. ასეთ სპეციალურ მეხსიერებას ეწოდება მაღაზიური მეხსიერება ან რასაც პროგრამისტები მეორენაირად უწოდებენ სტეკს. სტეკში სიმბოლოს ჩაწერა ხდება სტეკის თავში და სტეკიდან სიმბოლოს აღება ხდება იგივე სტეკის თავიდან. სტეკის თავი ყოველთვის უთითებს ბოლოს ჩაწერილ სიმბოლოს. როცა ვწერთ სიმბოლოს სტეკში, სტეკის თავი ერთი ადგილით გადაიწევს წინ და იქ ჩაიწერება მოცემული სიმბოლო ე. ი. წინათ ჩაწერილი სიმბოლოს წინ, ხოლო როცა ვიღებთ სიმბოლოს სტეკიდან მაშინ ხდება პირიქით. სტეკის თავი დაიწევს ერთი სიმბოლოთი უკან და სტეკის თავში მოხვდება შემდეგი სიმბოლო. ამგვარად, სტეკის თავი მოძრაობს წინ და უკან მასში სიმბოლოს ჩაგდება ამოგდების მიხედვით. ეს კი უზრუნველყოფს სტეკში ბოლოს ჩაგდებული სიმბოლოს პირველად ამოღებას. ასეთ სტეკებს უწოდებენ LIFO(Last Input First Output) სტეკებს. იმისათვის, რომ ისინი განვასხვაოთ FIFO(First Input First Output) სტეკებისაგან, რომელთა საშუალებით ხდება რიგების მოდელირება კომპიუტერში. ზოგადად, ავტომატი მაღაზიური მეხსიერებით შედგება:

1. შესასვლელი ფირისაგან, საიდანაც ავტომატი კითხულობს განსახილველი სიტყვის სიმბოლოებს, ისევე როგორც სასრული ავტომატის შემთხვევაში;

2. მართველი მოწყობილობისაგან და

3. სასრული მაღაზიური მეხსიერებისაგან ანუ LIFO სტეკისაგან (იხ. დიაგრამა):



დიაგრამა 2.6.1

ფორმალურად ავტომატი მაღაზიური მეხსიერებებით განიმარტება როგორც შვიდეული:

AM=(Q, Σ , δ , Γ, q_0, Z_0,F), სადაც

Q მდგომარეობათა სიმრავლეა,

Σ შესასვლელი სიმბოლოების სიმრავლეა,

Γ მაღაზიურ სიმბოლოთა სიმრავლეა,

$q_{0\in} Q$ საწყისი მდგომარეობაა,

z_0 მაღაზიის ძირში მოთავსებული სიმბოლოა, რომელიც მიუთითებს მაღაზიის დაცარიელებას. $F \subseteq Q$ საბოლოო მდგომარეობათა სიმრავლეა და δ არის $Q\times(\{\Sigma \cup e\})\times\Gamma$ სიმრავლის გადასახვა ისეთ სიმრავლეში, რომლის ელემენტებია $Q\times\Gamma^*$ სიმრავლის ქვესიმრავლეები.

სასრული ავტომატების ანალოგიურად განისაზღვრება AM ავტომატის კონფიგურაცია და ტაქტი. (q,W,d) არის AM ავტომატის მიმდინარე კონფიგურაცია, თუ ავტომატი იმყოფება q მდგომარეობაში, შესავალი სტრიქონია ამ მომენტში W და α არის მაღაზიის შიგთავსი. აქედან გამომდინარე (q,w,α) სამეული არის $Q\times\Sigma^*\times\Gamma^*$ ის ელემენტი. როცა $\alpha = e$ მაღაზია არის ცარიელი. დამოკიდებულებას $(q,aw,y\alpha) \vdash (q_1,W,\gamma\alpha)$ ეწოდება AM ავტომატის ტაქტი, როცა AM ავტომატი $(q,aw,y\alpha)$ კონფიგურაციიდან გადადის $(q,w,\gamma\alpha)$ ტაქტში და $\delta(q,a,y)$ სიმრავლე შეიცავს (q_1,γ) ელემენტს. ეს ნიშნავს, რომ AM ავტომატის მიმდინარე მდგომარეობაა q და შესასვლელის მიმდინარე სიმბოლოა a, მაღაზიის თავში გვაქვს y სიმბოლო და თუ $\delta(q,a,y)$ სიმრავლე შეიცავს (q,γ)-ს, მაშინ AM ავტომატის მიმდინარე მდგომარეობა გახდება q_1, მიმდინარე შესასვლელი სიმბოლო იქნება a-ს მომდევნო სიმბოლო მარცხნიდან მარჯვნივ მიმართულებით და მაღაზიაში y სტრიქონი შეიცვლება γ-თი. თუ $a=e$ ასეთ შემთხვევაში ტაქტს ეწოდება ცარიელი ტაქტი. ცარიელი ტაქტი შესაძლებელია მაშინაც, როცა შესასვლელი სტრიქონი ცარიელია, ხოლო თუ მაღაზია ცარიელია ე.ი. მაღაზიაში გვაქვს მხოლოდ z_0 სიმბოლო, შემდგომი ტაქტი შეუძლებელია. $\vdash^i$-ით აღინიშნება $\vdash$ დამოკიდებულების i ხარისხი ე.ი. მიმდევრობით შესრულებული ტექსტების i რაოდენობა, ხოლო $\vdash^*$ და $\vdash^*$ აღინიშნებიან შესაბამისად $\vdash$ დამოკიდებულების ტრანზიტულ- რეფლექსური და ტრანზიტული ჩაკეტვები. (q_0,w,z_0) არის საწყისი კონფიგურაცია, სადაც $W\in\Sigma^*$, ხოლო საბოლოო კონფიგურაციაა (q,e,α), თუ $q\in F$ და $\alpha\in\Gamma^*$. თუ $(q_0,W,z_0)\vdash(q,e,\alpha)$ რაიმე $q\in F$ და $\alpha\in\Gamma^*$, მაშინ ვიტყვით, რომ W სტრიქონს უშვებს AM ავტომატი ან რაც იგივეა $W\in L(AM)$ ე.ი. W ეკუთვნის AM ავტომატით განსაზღვრულ ენას და L(AM) არის AM ავტომატით დაშვებული ყველა სტრიქონის სიმრავლე.

განვიხილოთ მაგალითი. ავაგოთ ისეთი AM ავტომატი, რომელიც განსაზღვრავს $L=\{a^n b^{n|}\ n\geq 1\}$ ენას. იმისათვის, რომ ასეთი სახის ნებისმიერი სტრიქონი დაუშვას ავტომატმა საჭიროა შესასვლელი სტრიქონიდან თითო-თითოდ ჩავაგდოთ a ნიშნები სანამ არ ამოვწურავთ მათ და შემდეგ, ყოველი b-სთვის ამოვაგდოთ სიტყვიდან a ნიშანი და თუ შესასვლელი სტრიქონის დაცარიელების დროს

სტეკის თავში აღმოჩნდება z_0 სიმბოლო ამოვაგდოთ ისიც და ამ დროს AM ავტომატი უნდა იყოს საბოლოო მდგომარეობაში. რადგან სტეკი დაცარიელდება ავტომატს აღარ შეეძლება ახალ მდგომარეობაში გადასვლა და ამავე დროს შესასვლელი სტრიქონიც დაცარიელებულია. რაც იმას ნიშნავს, რომ ასეთ შესასვლელ სტრიქონს უშვებს AM ავტომატი. შევნიშნოთ, რომ ამ ენის დაშვება არ შეუძლია სასრულ ავტომატს, რადგან მას არა აქვს a ნიშნების დამახსოვრების საშუალება. ამგვარად, AM ავტომატს შეიძლება ჰქონდეს ასეთი სახე:

$AM=(\{q_0,q_1,q_2,\}, \{a,b\},\{a,z_0\},\delta ,q_0,z_0,\{q_2\})$, სადაც

$\delta\ (q_0,a,z_0,)=\{(q_1,az_0)\}$

$\delta\ (q_1,a,a)=\{(q_1,aa)\}$

$\delta\ (q_1,b,a)=\{(q_2,e)\}$

$\delta\ (q_2,b,a)=\{(q_2,e)\}$

$\delta\ (q_2,e,z_0)=\{(q_2,e)\}$

δ სიმრავლის პირველი ელემენტი უზრუნველყოფს ავტომატის საწყის მდგომარეობაში პირველი შესასვლელი a ნიშნის ჩაგდებას სტეკის ძირში და ავტომატი გადადის q_1 მდგომარეობაში. მეორე ელემენტი უზრუნველყოფს q_1 მდგომარეობაში მყოფი ავტომატისთვის შემდეგი a ნიშნის ჩაგდებას სტეკში. მესამე ელემენტი უზრუნველყოფს q_1 მდგომარეობაში მყოფი ავტომატისათვის შესასვლელი b ნიშნის შემთხვევაში სტეკიდან a ნიშნის ამოგდებას, როცა სვეტის თავში გვაქვს a ნიშანი და ავტომატი გადადის q_2 მდგომარეობაში. მეხუთე ელემენტი უზრუნველყოფს z_0 ამოგდებას სტეკის თავიდან, როცა ავტომატი q_2 მდგომარეობაშია შესასვლელი ნიშნის გამოუყენებლად და ამგვარად, აცარიელებს სტეკს. თუ ამ მომენტში შესასვლელი სტრიქონი ცარიელია, მაშინ შესასვლელ სტრიქონს უშვებს AM ავტომატი. ყველა შემთხვევაში, გარდა ბოლო შემთხვევისა, შესასვლელ სტრიქონში მიმდინარე ნიშანი ხდება შემდეგი ნიშანი. განვიხილოთ, რაიმე შესასვლელი სტრიქონი $w\in\{a,b\}^*$ და ვთქვათ, AM ავტომატი (q_0,w,z_0) საწყის მდგომარეობაშია. AM ავტომატმა, რომ გააგრძელოს მუშაობა, ე.ი. გადავიდეს შემდეგ კონფიგურაციაში, მას აქვს ერთად-ერთი შესაძლებლობა გამოიყენოს δ სიმრავლის პირველი ელემენტი. აგრამ, ამისათვის საჭიროა $w=aw_1$. ასეთ შემთხვევაში, ავტომატი გადადის ახალ კონფიგურაციაში (q_1,w_1,az_0). ამ კონფიგურაციაში MA ავტომატს შეუძლია გამოიყენოს δ-ს მეორე ან მესამე ელემენტი. მეორე ელემენტის გამოყენების შემთხვევაში, უნდა გვქონდეს $w_1=aw_2$ და MA ავტომატი

გადავა კონფიგურაციაში (q_1,w_2,aaz_0) და ჩვენ ვართ იგივე სიტუაციაში, რაც წინა კონფიგურაციაში. ე.ი. ჩვენ კვლავ შეგვიძლია გამოვიყენოთ ბ სიმრავლის მეორე ან მესამე ელემენტი. ვთქვათ, გამოვიყენეთ ბ სიმრავლის მეორე ან მესამე ელემენტი. ვთქვათ, გამოვიყენეთ ბ-ს მეორე ელემენტი k-ჯერ, მაშინ გვექნება ასეთი კონფიგურაცია (q_1,w_{k+2}, $a^{k+2}z_0$). ამის შემდეგ, MA რომ მოხვდეს q_2 მდგომარეობაში

PTQ ფრაგმენტი

ამ თავში წარმოდგენილია ინგლისური ენის ფრაგმენტი, რომელიც აღწერილია მონტეგიუს შრომაში PTQ. ვიწყებთ პატარა ფრაგმენტით და მას ვაფართოვებთ თანდათანობით. ყოველ ნაბიჯზე სინტაქსი და სემანტიკა გაანალიზებულია ფართოდ. სპეციალური ყურადღება ექცევა მოტივაციას და ანალიზის დაზუსტებას. სპეციალური ყურადღება მიექცევა აგრეთვე დამუშავების ალგებრულ და ალგორითმულ ასპექტებს. ალგებრული განხილვა ითვალისწინებს იმის ახსნას, თუ რატომ ზოგიერთი დეტალები არიან ასე და არა სხვანაირად. ალგორითმული ასპექტები ეხება იმ მეთოდს, რომლის საშუალებითაც მარტივად წარმოიდგინება მნიშვნელობები. იმსათვის რომ მივაღწიოთ ამას ზოგიერთი წესები მოცემული იქნება ფორმულების გასამარტივებლად. ფრაგმენტი მდიდარია სემანტიკურად საინტერესო მოვლენებით. ქვემოთ მივუთითებთ ზოგიერთ მათგანს კომენტარებით.

მოვლენის პირველი სახეობა, რომელიც ეხება ისეთ წინადადებებს, რომელთა აზრი ნათელია და განიხილება, თუ როგორ უნდა წარმოვადგინოთ ისინი სტანდარტული პრედიკატული ლოგიკის გამოყენებით.

განვიხილოთ (1) და (2)

(1) John runs.

(2) Every man runs.

ეს წინადადებები გარეგნული ფორმით ჰგვანან ერთი მეორეს. მხოლოდ ქვემდებარით განსხვავდებიან. მაგრამ, მათი აზრების წარმოდგენები არიან განსხვავებული. სტანდარტულ ლოგიკაში მათი აზრების წარმოდგენებია:

(3) run (John)

(4) $\forall$ X [man (x)$\rightarrow$ run (x)] .

ეს გვაძლევს გამოსავალს იმისა, თუ როგორ მივიღოთ განსხვავებული ფორმულები მსგავსი წინადადებებისათვის. ანალოგიური საკითხი წარმოიშვება (5)-ის ომონიმიურობისას.

(5) Every man loves a woman.

ეს წინადადება შეიძლება იყოს გაგებული როგორც კონკრეტული რომელიმე ქალი შეყვარებულია ყოველი მამაკაცის მიერ (მაგალითად, Brigitte Bardot), ან როცა ყოველი მამაკაცისათვის შეიძლება იყოს სხვადასხვა ქალი (ვთქვათ, თავისი საკუთარი დედა). ამგვარად, (5) არის ომონიმიური. ამ ორი შესაძლებლობის გამო, ასეთი სახის ომონიმიას ეწოდება "საზღვრების ომონიმია" ქვანტიფიკატორებისათვის. ორი სხვადასხვანაირი წაკითხვა ამ წინადადებისა გრამატიკულად მოცემულია (6)-სა და (7)-ში.

(6) $\forall$ X [man(x)$\rightarrow \exists$ y[woman(y) $\wedge$ love(x,y)]]

(7) $\exists$ y [woman(y) $\wedge \forall$ x[man(x)$\rightarrow$ love(x,y)]] .

მოვლენის მეორე სახეობა ეხება წინადადებებს, რომელთათვისაც ძნელია იმის თქმა, თუ როგორ უნდა იყოს წარმოდგენილი მათი მნიშვნელობები. განვიხილოთ (8) და (9)

(8) John seeks a unicorn.

(9) John finds a unicorn.

ამ ორ წინადადებას აქვს ერთნაირი ფორმა და მხოლოდ მათი ზმნები განსხვავდებიან ერთიმეორისაგან. ვინმემ შეიძლება იფიქროს რომ მათ უნდა ჰქონდეთ ერთნაირი აზრები აგრეთვე. ერთადერთი განსხვავება არის ის, რომ სხვადასხვა დამოკიდებულებებს გამოხატავენ ჯონსა და მარტორქას შორის. მაგრამ ეს ასე არაა. მიდგომა, რომელიც ამბობს, რომ seek-დამოკიდებულება არის ყოველთვის რაიმე დამოკიდებულება ორ ინდივიდს შორის არ უნდა იყოს მისაღები. ჩვენ უნდა გავითვალისწინოთ (8)-სთვის რაიმე აზრი, რომლიდანაც არ გამომდინარეობს, რომ მარტორქები არსებობენ. მაგრამ (8) შეიძლება იყოს გამოყენებული იმ სიტუაციაშიც, როცა მარტორქები არსებობენ. ამგვარად, გვაქვს ომონიმია ამ ორ შესაძლებლობას შორის. მას აქვს წაკითხვა, რომლიდანაც გამომდინარეობს, რომ ყველაზე მცირე ერთი მარტორქა მაინც არსებობს (რეფერენციალური წაკითხვა) და წაკითხვა, რომლიდანაც ეს არ გამომდინარეობს (არარეფერენციალური წაკითხვა). რეფერენციალური და არარეფერენციალური წაკითხვის სხვა მაგალითებია (10), (11) და (12).

(10) John talks about a unicorn. (ჯონი საუბრობს მარტორქის შესახებ)

(11) John wishes to find a unicorn and eat it.(ჯონს სურს იპოვოს მარტორქა და შეჭამოს იგი)

(12) Mary believes that John finds a unicorn and that he eats it.(მერის სჯერა, რომ ჯონი იპოვის მარტორქას და შეჭამს მას)

მე-(9) წინადადება უზრუნველყოფს მხოლოდ რეფერენციალურ წაკითხვას. იგივეს ადგილი აქვს (13)-ისთვის.

(13) John tries to find a unicorn and wishes to eat it.(ჯონი ცდილობს იპოვოს მარტორქა და შეჭამოს იგი)

ომონიმია, რომელსაც ჩვენ ვასხვავებთ (8), (9), (11) და (12) წინადადებებში ლიტერატურაში ცნობილია როგორც "de-dicto/de-re" ომონიმია ან ძ"ძსპეციფიკური / არასპეციფიკური" ომონიმია.

2. John runs (ჯონი მირბის)

ეს ფრაგმენტი შედგება ძალიან მარტივი წინადადებებისაგან. როგორიცაა, John runs. გვაქვს სამი კატეგორია (სორტი): თერმების კატეგორია T, გარდაუვალი ზმნური ფრაზების კატეგორია IV და წინადადებების კატეგორია S. არსებობენ ძირითადი ბაზისური გამოსახულებები (გენერატორები) T და IV კატეგორიებისათვის. T კატეგორიის B_T გენერატორების სიმრავლე შეიცავს PTQ ფრაგმენტის საკუთარ სახელებს. შემდგომში მათ დაემატება სპეციალური სახელი საილუსტრაციოდ Bigboss. სიმრავლეები B_T და B_{IV} განისაზღვრებიან ასე:

2.1. B_T={ John, Bill, Mary, Bigboss}
2.2. B_{IV}={ run, walk, talk, rise, change}.
2.3. End

ლოგიკაში B_T-ს ელემენტებს შეესაბამება e ტიპის კონსტანტები, Bigboss-ის გარდა. ინტენსიონალურ ლოგიკაში e ტიპის კონსტანტებს შორის ჩვენ განვასხვავებთ სამ სპეციალურ მათგანს:

2.3. {John, bill, mary} $\subset CON_e$
2.3.end
$C^{A,i,g}$=F(C) (i) (C$\in$ CON_e)

2.4. მნიშვნელობის 1 პოსტულატი:
$\exists$ u □ [u = α] სადაც $\alpha \in$ { John, bill, mary}
2.4. end

2.5. განსაზღვრება. [α / Z] $\varnothing$ აღნიშნავს ყველა
თავისუფალი Z -ის ჩანაცვლებას α -თი Φ -ში.

2.6. თეორემა λ U [Φ] α = [α /] Φ სადაც $\alpha \in$ { John, bill, mary}

2.7. I L ფორმულას ეწოდება მოდალურად ჩაკეტილი
თუ იგი არის შემდეგი I L ქვეალგებრის ელემენტი:
< [{ John, mary, bill}] , (VARτ) $\tau \in$ T Y , R$\cup$ { R$\wedge$, R□ } >

2.8. გამარტივების წესი

1. ვთქვათ Z$\in$ VARτ $_1$, $\alpha \in$ MEτ $_2$ და $\beta \in$ MEτ 2

შემდეგ, შევცვალოთ λ Z [β] (α) [α / Z] β თუ

1) β -ს არცერთი ცვლადი არ ხდება დაბმული ამ ჩასმის შედეგად და ან

2) Z არაა დაბმული β -ში ∧ , H , W ან □ საზღვრებში ან

3) β არის მოდალურად ჩაკეტილი.

Bigross არ შეიძლება ითარგმნოს როგორც e ტიპის ხისტი კონსტანტა. ჩვენ შეგვიძლია იგი ვთარგმნოთ <s,e> ტიპის კონსტანტაში, ანდა e ტიპის რაიმე კონსტანტაში ვთარგმნოთ და შემდეგ მას გავუკეთოთ ინტერპრეტაცია არახისტად. ჩვენ ავირჩევთ პირველ გზას.

2.9. bigboss ∈ CON< s, e>

2.9. end.

bigboss კონსტანტის ინტერპრეტაცია არის ფუნქცია ინდექსიდან ინდივიდუმში. ასეთ ფუნქციას ეწოდება ინდივიდუალური ცნება (კონცეპტი). ამგვარად ∧ John აღნიშნავს ინდივიდუალურ ცნებას. ∧ John-ით აღნიშნული ინდივიდუალური ცნება არის კონსტანტა ფუნქცია, მაშინ, როცა bigboss არაა კონსტანტა ფუნქცია.

ვთქვათ ძალთა ბალანსი იცვლება და ბრეჟნევი ხდება bigboss რეიგანის ნაცვლად. ეს შეიძლება გამოისახოს წინადადებით (14).

(14) Bigboss changes.

(14) აზრი არ იქნება სწორად წარმოდგენილი ფორმულით, რომელიც ამბობს, რომ change პრედიკატი გამოიყენება რაღაცა ინდივიდუმზე. შეიძლება შეიცვალა აბსოლუტური ძალა რეიგანისა (შემცირდა) ან შეიცვალა აბსოლუტური ძალა ბრეჟნევისა (გაიზარდა). ანდა შეიძლება ორივეს ძალა შეიცვალა. წინადადება (14) ამბობს რომ ცნება 'Bigboss' შეიცვალა იმ აზრით, რომ იგი ეხება სხვა პიროვნებას. ამგვარად, (14)-ის აზრი შეიძლება იყოს წარმოდგენილი - ფორმულით, რომელიც ამბობს, რომ პრედიკატ changes-ს ადგილი აქვს ინდივიდუალური ცნებისათვის, რომელიც დაკავშირებულია Bigboss-თან. ასეთი ანალიზისათვის change უნდა იყოს

< < s,e> , t > ტიპის. სინტაქსსა და სემანტიკას შორის ჰომომორფული

დამოკიდებულების გამო, ეს ნიშნავს, რომ ყველა გარდაუვალი ზმნები < < s,e>, t> ტიპის უნდა იყოს. ამგვარად,

2.10. { run, walk, rise, change} ⊂ CON< < s,e> ,t>
2.11. run ′ =run, walk′ =walk, talk ′ = talk, rise ′ =rise, change′ =change

2.11. end

განსაზღვრულ ფრაზიანი გრამატიკები (DCG)

როგორც ცნობილია, ბუნებრივი ენის ძირითადი სინტაქსური ანალიზი შეიძლება აღიწეროს კონტექსტისაგან თავისუფალი გრამატიკით (CFG). კონტექსტისაგან თავისუფალი გრამატიკათა შემდგომ განზოგადოებას წარმოადგენს განსაზღვრულ ფრაზიანი გრამატიკები. ასეთ გრამატიკაში არატერმინალურ სიმბოლოდ შეიძლება ავიღოთ ისეთი გრამატიკული კატეგორიები, როგორიცაა წინადადება, ქვემდებარის ჯგუფი, ზმნის ჯგუფი, დამატების ჯგუფი და სხვა. შეგვიძლია განვმარტოთ, რომ წინადადება შედგება ქვემდებარის ჯგუფისაგან, ზმნის ჯგუფისაგან და ერთი ან ორი დამატების ჯგუფისაგან. თუ ჩვენ გავაგრძელებთ თვითეული ჯგუფის განმარტებას დავინახავთ, რომ ქვემდებარის ჯგუფი და დამატების ჯგუფები თავიანთი ზოგადი აღნაგობით არ განსხვავდებიან ერთიმეორესაგან. ამიტომ, მათთვის შეიძლება შემოვიღოთ ერთი საერთო სახელი - სახელის ჯგუფი და იგი განვმარტოთ. მეორეს მხრივ, წინადადებაში რამდენი სახელის ჯგუფია დამოკიდებულია ზმნის ჯგუფზე. ამიტომ, მიზანშეწონილია წინადადების განმარტება დავუკავშიროთ ზმნის ჯგუფის განმარტებას და ზმნის ჯგუფის მიხედვით შემოვიტანოთ სახელთა ჯგუფები წინადადებაში და ეს პროცესი გავაგრძელოთ მანამ, სანამ არ დავალთ ისეთ კატეგორიებამდე როგორიცაა მეტყველების ნაწილები. მაგალითად, არსებითი სახელები, ზედსართავები, კავშირები, ზმნისზედები და სხვა. მათი განმარტება შეიძლება მათში შემავალი ელემენტების (სიტყვების) ჩამოთვლით, რომლებიც შემდგომ განმარტებას არ ექვემდებარებიან. მათ ეწოდებათ ტერმინალური სიმბოლოები. ყოველი განსამარტავი გრამატიკული კატეგორია ავღნიშნოთ რაიმე სიმბოლოთი. მაგალითად წინადადება - S-ით, სახელის ჯგუფი - P-თი და ზმნის ჯგუფი - VP-თი. მაშინ, წესი S→ VP აღნიშნავს რომ წინადადება არის ზმნის ჯგუფი, ხოლო წესი VP→ np& vp1& np (1) აღნიშნავს, რომ ზმნის ჯგუფი შედგება სუბიექტის ჯგუფისაგან, ორადგილიანი პრედიკატის და პირდაპირი ობიექტის ჯგუფისაგან სიღრმისეულ დონეზე, ხოლო ზედაპირულ დონეზე გამოსახულია სახელის ორი ჯგუფით და ზმნის ჯგუფით, რომელთაც გააჩნიათ თავიანთი დამახასიათებელი ფორმალური ნიშნები, რომლებიც განსაზღვრავენ გრამატიკულად სწორად შედგენილ წინადადებას, ხოლო სიღრმისეულ დონეზე ჯგუფები უნდა იყვნენ ერთიმეორესთან თავსებადნი, რომ მოგვცეს აზრობრივად სწორი წინადადება. თავსებადობა განისაზღვრება ჯგუფის სემანტიკური ნიშნებით და გამოისახება სემანტიკური წესის საშუალებით. ამგვარად, წინადადების გრამატიკულად სწორად გაფორმება გამოისახება სინტაქსური წესებით, ხოლო მის აზრობრივ სისწორეს განსაზღვრავს შესატყვისი სემანტიკური წესი. ჩვენ აქ არ შევეხებით სემანტიკურ წესებს და დავკმაყოფილდებით მხოლოდ სინტაქსური წესების განხილვით. დავუბრუნდეთ (1) წინადადებას. იმისათვის, რომ გამოვსახოთ სინტაქსურად სწორი წინადადების კერძო სახეობა, დავაზუსტოთ იგი იმ ნიშნებით, რომლებიც მოითხოვება ორპირიანი გარდამავალი ზმნის ფორმებისათვის. ეს ნიშნები კარგადაა ცნობილი ქართული გრამატიკის სახელმძღვანელოებში. კონკრეტულად, თუ ზმნის ფორმა პირველი სერიიდანაა, მაშინ პირველი სახელის ჯგუფის მთავარი სახელი უნდა იყოს სახელობით ბრუნვაში (შეესაბამება სუბიექტის ჯგუფს სიღრმისეულ

დონეზე), ხოლო მეორე სახელის ჯგუფის მთავარი სახელი უნდა იყოს მიცემით ბრუნვაში (შეესაბამება პირდაპირი ობიექტის ჯგუფს სიღრმისეულ დონეზე). ანალოგიურად ჩამოყალიბდება წესები ზმნის ფორმებისათვის მეორე და მესამე სერიებიდან. ყველა შემთხვევაში ზმნის ფორმა უნდა იყოს შეთანხმებული რიცხვში სუბიექტის ჯგუფთან და პირში სუბიექტისა და ობიექტის ჯგუფებთანაც. ამავე დროს უნდა იყოს გათვალისწინებული, რომ ხსენებული ჯგუფები შეიძლება იყვნენ დალაგებულნი წინადადებაში ნებისმიერად. ახლა, ვნახოთ თუ როგორ შეიძლება სინტაქსური წესები გამოვსახოთ DCG-ში და შემდეგ როგორ გადაიწერებიან ისინი პროლოგზე.

S(X) პრედიკატით აღვნიშნოთ ის ფაქტი, რომ X არის სია, რომლის ელემენტებია რაიმე წინადადების სიტყვები, იგივე მიმდევრობით, რომლითაც ისინი გვხვდებიან წინადადებაში, ე. ი. P(X) არის ერთარგუმენტიანი პრედიკატი, სადაც X არის სია და სიის ელემენტები არიან სტრიქონული ტიპის კონსტანტები ან ცვლადები. ანალოგიურად, NP(X) გამოსახავს სახელის ჯგუფს და VP(X) ზმნის ჯგუფს. ამგვარად, ზემოთ მოცემული წესები შეიძლება გამოვსახოთ ჰორნის წინადადებებით შემდეგნაირად:

S(X)← VP(X).

VP(X)← append(X1,X4,X)& append (X2,X3,X4,)& (2)

np| (X1)& VP1(X2)& np2(X3)&

აქ, append (X1,X2,X3) აღნიშნავს X3 სიის გახლეჩას X1 და X2 სიებად, სადაც X1 და X2 სიების ელემენტთა კონკატენაცია იგივეა რაც X3-ის ელემენტთა კონკატენაცია. ზემოთ მოცემული (2) წინადადებები ჩვენ შეგვიძლია გადავწეროთ უფრო მოხერხებული სახით. ამისათვის, რაიმე X სია ჩვენ შეგვიძლია წარმოვიდგინოთ ორ სიად Y და Z, სადაც X არის Y და Z-ის კონკატენაცია. კერძოდ, Z შეიძლება იყოს ცარიელი სია, რომელიც აღვნიშნოთ nil-ით. ამგვარად, S(X) პრედიკატი შეიძლება შევცვალოთ ახალი პრედიკატით S(Y,Z), ზემოთ მიღებული შეთანხმების გათვალისწინებით. ამის შემდეგ, (2) წინადადებები გადაიწერებიან ასე:

S(X,Z)← VP(X,Z).

VP(X,V)← np1(X,Y)& VP1(Y,Z)& np2(Z,V). (3)

განსახილავი წინადადება ჩვენ შეგვიძლია მივაწოდოთ პროლოგ პროგრამას მიზნის საშუალებით. ასე, მაგალითად,

← S(პეტრე, აშენებს, ახალ, სახლს, nil, nil)

თუ ჩვენ გამოვიყენებთ მაკკორდის მიერ შემოტანილ აღნიშვნებს – – > და : , სადაც ისარი არის ოპერატორი, რომელიც აცალკევებს წესის დასათაურებას წესის

ტანისაგან, ხოლო : ოპერატორი გამოჰყოფს ტანის ელემენტებს ერთიმეორესაგან, მაშინ (3) წინადადებები გადაიწერება ასე:

S– – > VP.

VP– – > np1 : VP1 :np2. (4)

(4)-ში S, np, np1 და np2 არიან არატერმინალური სიმბოლოები და წარმოადგენენ (3)-ში მოცემული პრედიკატების შემოკლებულ ჩაწერას. ახლად შემოტანილი ოპერატორებისათვის უნდა დადგინდეს მათი პრიორიტეტები. ცხადია :-ს უნდა ჰქონდეს უფრო მაღალი პრიორიტეტი ვიდრე – – > -ს. IBM პროლოგში ოპერატორის პრიორიტეტი მოიცემა ჩაშენებული პრედიკატით OP. მაგ. OP("– – > ″ , rl, 20), სადაც rl აღნიშნავს ორ ოპერანდიან მარჯვნივ ასოციატიურ ოპერატორს. არატერმინალები განსაზღვრავენ სიტყვების სიათა კლასებს, ხოლო ტერმინალები განსაზღვრავენ ცალკეულ სიტყვებს. ტერმინალები გამოისახებიან + T ფორმით, სადაც + არის პრეფიქსული ოპერატორი და უნდა ჰქონდეს სხვა ოპერატორებზე უფრო მაღალი პრიორიტეტი. ზემოთ მიღებული შეთანხმებებით ახლა შევადგინოთ მარტივი გრამატიკა, რომელიც გამოდგება ″პეტრე აშენებს ახალ სახლს″-ის მსგავსი წინადადებების გამოცნობისათვის:

s – – > p

VP – – > np1:vp1:np2.

vp1– – > v

np1– – > პეტრე

np2 – – > ad : N; N.

v – – > + აშენებს.

N – – > +სახლს.

ad – – > + ახალ.

წინა ლექციაში განხილული გრამატიკა ჩვენ შეგვიძლია იოლად გადავიყვანოთ ჰორნის წინადადებებში თუ გავიხსენებთ, რომ ყოველი არატერმინალი nt გადაიყვანება ორადგილიან პრედიკატში nt(x,y) და : ოპერატორს შევცვლით & -ით. + t, სადაც t არის ტერმინალური სიმბოლო, უნდა გადავიყვანოთ პირობაში x=t, y წესში მისი პოზიციის გათვალისწინებით. მაგალითად, წესი

vp– – > np : vp1 : + ახალ : n, გადაიყვანება წინადადებაში

vp(x,v)← np(x,y)& vp1(y,z)& z= ახალ.u& n(u,v)

შესაძლებელია ამ წესის შემდგომი გამარტივება თუ z-ს შევცვლით მისი მნიშვნელობით:

vp(x,v)← np(x,y)& vp1(y,ახალ.u)& n (u,v).

ასევე წესი n– – > + სახლს გადაიყვანება წინადადებაში

n(x,y)← x= სახლს. y.

ხოლო X-ის გამორიცხვით მიიღება

n(სახლს.y,y).

ამგვარად, (5) გრამატიკა გადაიყვანება შემდეგ პროლოგ პროგრამაში:

s(x,y)← vp(x,y).

vp(x,u)← np1(x,y)& vp1(y,z)& np2(z,u).

vp1(x,y)← v(x,y).

np2(x,z)← ad(x,y)& n(y,z).

np2(x,y)← n(x,y).

np1(პეტრე.x,x).

v(აშენებს.x,x).

n(სახლს.x,x).

ad(ახალ.x,x).

მიღებული პროლოგ პროგრამა შეიძლება გამოყენებულ იქნეს როგორც (5) გრამატიკით განსაზღვრული წინადადებების გამოსაცნობად, ასევე, (5)-ით განსახღვრული წინადადებების გენერირებისათვის. თუ (6) პროგრამას მივცემთ მიზანს ← s(x, nil)& write (x) & fail. პასუხად მივიღებთ ყველა წინადადებას განსაზღვრულს (5) გრამატიკით, ხოლო თუ მივცემთ მიზანს

← S(პეტრე. აშენებს. ახალ. სახლს, nil).

მოგვცემს დადებით პასუხს, რაც ნიშნავს რომ "პეტრე აშენებს ახალ სახლს" წინადადება ეკუთვნის (5) გრამატიკით განსაზღვრულ ენას. როგორც ადვილი შესამჩნევია, ასეთი გრამატიკა პრაქტიკულად უვარგისია მისი შემდგომი გაფართოების გარეშე. პირველ რიგში, ძალზე მოუხერხებელია პრაქტიკული რეალიზაციის თვალსაზრისით ქართული ენის სიტყვის ფორმების ტერმინალურ სიმბოლოებად გამოცხადება, რაც იმას ნიშნავს, რომ სიტყვათა ყველა ფორმები უნდა შევიტანოთ კომპიუტერულ ლექსიკონში. ამიტომ, ამ ეტაპზე შეიძლება ვიგულისხმოთ, რომ ყოველი სიტყვა შეიძლება დახასიათებულ იქნეს მისი უცვლელი ნაწილით (ლექსიკური მნიშვნელობით) და იმ მორფოლოგიური კატეგორიებით, რომლებიც აღწერენ სრულად ამ სიტყვას. ჩვენ ვიგულისხმებთ, რომ სიტყვის ფორმა შეცვლილია მისი უცვლელი ნაწილით და მორფოლოგიური კატეგორიებით, რომლებიც ამ სიტყვას აღწერენ. ამისათვის, შემოვიტანთ აქ ჩვენ სათანადო პრედიკატებს, რომლებსაც არგუმენტებად ექნებათ საჭირო მორფოლოგიური კატეგორიები და ასევე სხვა ინფორმაციაც. მაგალითად, არსებითი სახელებისათვის შეიძლება შემოვიტანოთ პრედიკატი არს-სახ, მაშინ იმისათვის, რომ მივუთითოთ სიტყვის ფორმა სახლს საჭიროა არს-სახ პრედიკატს ჰქონდეს არგუმენტები, რომლებიც გვიჩვენებენ ბრუნვას, რიცხვს და სიტყვის უცვლელ ნაწილს. კერძოდ თუ დავწერთ არს-სახ (სახლ, მიც, მხ), სადაც პირველი არგუმენტი მიუთითებს სიტყვის უცვლელ ნაწილს, მეორე არგუმენტი ბრუნვას და მესამე არგუმენტი რიცხვს და ჩანაწერები: სახლ, მიც და მხ არიან არგუმენტების მნიშვნელობები. სახლ წარმოადგენს სიტყვის "სახლს" უცვლელ ნაწილს, ხოლო მიც და მხ წარმოაგენენ მიცემითი ბრუნვისა და მხოლობითი რიცხვს აღნიშვნებს შესაბამისად. როგორც მაგალითიდან ჩანს, პრედიკატის დასახასიათებლად არაა საკმარისი მისი სახელისა და არგუმენტების რაოდენობისა და რიგის ცოდნა, ასევე საჭიროა თვითეული არგუმენტის ტიპის ცოდნა ანუ დავახასიათოთ თვითეული არგუმენტისათვის მისი მნიშვნელობათა სიმრავლე. მოყვანილ მაგალითში პრედიკატს არს-სახ აქვს სამი არგუმენტი, რომელიც საკმარისი იყო მოცემული თერმის "სახლს" დასახასიათებლად, მაგრამ ეს იმას არ ნიშნავს, რომ საზოგადოდ არსებით სახელთა დასახასიათებლად საკმარისი იქნება სამი არგუმენტი. არგუმენტთა რაოდენობა დამოკიდებულია იმაზე, თუ რას ვისახავთ მიზნად ამ

პრედიკატის შემოტანით. თუ ჩვენ ვისახავთ მიზნად, დავახასიათოთ ყველა არსებითი სახელები მორფოლოგიურ დონეზე, მაშინ არგუმენტების რაოდენობა იქნება ვთქვათ m, ხოლო თუ ასევე ვიგულისხმებთ, რომ ეს პრედიკატი შეიცავდეს ინფორმაციას არსებით სახელთა დასახასიათებლად სემანტიკურ დონეზეც, მაშინ არგუმენტების რაოდენობა იქნება უფრო მეტი. თუ ეს პრედიკატი გვჭირდება როგორც საწყისი ინფორმაცია არსებითი სახელის მორფოლოგიური კატეგორიების დასადგენად, მაშინ მას უნდა ჰქონდეს არგუმენტები, სიტყვის ფუძისათვის, ბრუნების ტიპისათვის და სხვა ისეთი ინფორმაციისათვის, რომელიც ჩვეულებრივ მოიცემა მანქანური თარგმნის კომპიუტერულ ლექსიკონებში არსებითი სახელის ფუძეებისათვის. ამგვარად, ტერმინალური სიმბოლოები შეიცვლება პრედიკატებით, რომლებიც იღებენ მნიშვნელობებს თავიანთი არგუმენტებისათვის ცოდნის ბაზიდან. ე.ი. ასეთი პრედიკატები უნდა იყოს განსაზღვრული ცოდნის ბაზაში. ასეთი პრედიკატების ცოდნის ბაზაში შეტანა ხდება უფრო დაბალი დონის ანალიზის დროს. მაგალითად, სინტაქსური ანალიზისათვის უფრო დაბალი დონის ანალიზი იქნება მორფოლოგიური ანალიზი. გარდა ამისა, იმისათვის, რომ წესებში მივუთითოთ ჯგუფებს შორის არსებული მოთხოვნების დაცვა, არატერმინალური სიმბოლოების შესაბამისი პრედიკატებისათვის უნდა შემოვიტანოთ კიდევ დამატებითი არგუმენტები. მაგალითად, არსებითი სახელის ჯგუფსა და ზმნის ჯგუფს შორის რიცხვში შეთანხმებისათვის და სხვა. ასეთნაირად გაფართოებული გრამატიკა უკვე გამოდგება იმისათვის, რომ დავადგინოთ რაიმე წინადადება არის თუ არა სინტაქსურად სწორად შედგენილი. იმისათვის, რომ ასეთმა გრამატიკამ მოგვცეს, აგრეთვე, მოცემული წინადადების სინტაქსური სტრუქტურა, საჭიროა ყოველ არატერმინალურ სიმბოლოს შესაბამის პრედიკატს დავუმატოთ ახალი არგუმენტი, რომელიც იქნება სია და დასაშვებია, რომ ამ სიის ელემენტები იყოს კვლავ სია და ა. შ..

ანალიზისა და გენერაციის ქვემოდან ზემოთ მიმართული ალგორითმი

ეს ალგორითმი არის ქვემოდან ზემოთ მიმართული ცხრილური ანალიზის ალგორითმი, რომლის ლექსიკური ერთეულების მოძებნის ფაზა დაზუსტებულია ისე, რომ იგი გამოდგეს გენერაციისათვისაც.

ძირითადი განსხვავება ანალიზსა და გენერაციას შორის არის შემდეგი:

1. ანალიზის დროს გამოიყენება ის სიტყვები, რომლებიც გვხვდება შესავალ სტრიქონში, მაშინ როცა გენერაციის დროს გამოიყენება ლექსიკონის ყველა ერთეულები.

2. ერთეულები, რომლებიც გამოიყენებიან ანალიზისას არიან დალაგებულნი შესასვლელ სტრიქონში მათი შეხვედრის რიგის მიხედვით, მაშინ როცა გენერაციის დროს ერთეულები შეიცავენ ინფორმაციას იმის შესახებ, თუ რომელ სტრიქონებში შეიძლება შეგვხვდნენ ისინი. აქედან გამომდინარე შეიძლება დავასკვნათ, რომ ანალიზი არის გენერაციის უფრო შეზღუდული სახეობა, რადგან ანალიზის დროს ცნობილია, რომელია ასაგები სტრიქონი (იხ. Kay, Martin. Algorithm Schemata and Data Structures in Sintactic Processing. Report CSL-8012, Palo Alto, CA: XEROX PARC). ამიტომ,

აგებულია ერთიანი ალგორითმი, როგორც გენერაციისათვის ასევე ანალიზისათვის, სადაც განსხვავება თავს იჩენს მხოლოდ ალგორითმის ინიციალიზაციის ფაზაში. ალგორითმი განასხვავებს აქტიურ და პასიურ ერთეულებს. რაიმე წესი არის გამოყენებული რაიმე ერთეულზე, თუ ერთეულის კატეგორია უკავშირდება წესის მარჯვენა მხარის პირველ კატეგორიას, მაშინ ახალი ერთეული, რომლის დაბოლოება არის წესის მარჯვენა მხარის დარჩენილი ნაწილი. თუ დარჩენილი ნაწილი არაა ცარიელი, მაშინ ერთეულს ეწოდება აქტიური ერთეული და მას შეუძლია მიუერთდეს რაიმე პასიურ ერთეულს, რომლის კატეგორია უკავშირდება დარჩენილი ნაწილის ყველაზე მარცხენა ელემენტის კატეგორიას. რაიმე ერთეული არის < Span, String, Category, Remainder> .

Span არის წყვილი < დასაწყისი ძირი, ბოლო ძირი> .

String არის სიტყვების სია.

Category არის გრამატიკის რაიმე არატერმინალური სიმბოლო.

Remander არის კატეგორიების სია. თუ სია ცარიელია მაშინ ერთეულს ეწოდება პასიური (დასრულებული), წინააღმდეგ შემთხვევაში იგი არის აქტიური (დაუსრულებელი). ერთეულები არიან აქტიურები ან დაკიდებულები. აქტიურ ერთეულთა სიმრავლეს ეწოდება ცხრილი, ხოლო დაკიდებულ ერთეულთა სიმრავლეს - ნიმუში. დაკიდებულ ერთეულებს აქვთ პრიორიტეტები და ისინი ემატებიან ცხრილს მათი პრიორიტეტების კლების მიხედვით. როცა დაკიდებული ერთეული ემატება ცხრილს, იგი ხდება აქტიური და ახალი დაკიდებული ერთეულები იქმნებიან რაიმე წესის გამოყენებით ერთეულზე ან ერთეულთა კომბინირებით, როცა ერთერთი აქტიურია და მეორე პასიური. ასეთი ალგორითმი საშუალებას იძლევა გამოვიყენოთ სხვადასხვა გარჩევის სტრატეგიები.

ხეთა შეერთების გრამატიკები (TAG)

ხეთა შეერთების გრამატიკა (Tree-Adjoining Grammar) გამიზნულია სტრიქონთა გენერაციის სისტემისათვის, უფრო ზუსტად, იმ სტრუქტურების გენერაციისათვის, რომლებიც წარმოდგენილი არიან ხეების საშუალებით. იმისათვის, რომ ავღწეროთ ხეების გამოყვანა ობიექტურ ენაზე, საჭიროა ეს ხეები გამოვიყვანოთ ობიექტური ენის ხეებისაგან. ასეთი გამოყვანები საინტერესოა, როგორც სინტაქსური - ასევე სემანტიკური თვალსაზრისით. TAG შემოტანილია E. Joshi-ის მიერ 1975 წელს ([Joshi 1975]) და საბოლოო სრული ვარიანტი აღწერილია Joshi-ის მიერ 1991 წ. ([Joshi 1991]). ხეთა შეერთების ენა (TAG) ეკუთვნის კონტექსტზე დამოკიდებულ ენებს, უფრო ზუსტად, შუალედურ კონტექსტის მგრძნობიარე ენათა კლასს.

ხეთა შეერთების გრამატიკა ეწოდება ხუთეულს (Σ , N , I , A , S), სადაც:

(1) Σ არის ტერმინალურ – სიმბოლოების სასრული სიმრავლე;

(2) N არის არატერმინალური სიმბოლოების სასრული სიმრავლე;

(3) S არის გამოყოფილი არატერმინალური სიმბოლო, $S \in N$;

(4) I არის სასრული ხეების სიმრავლე, რომელსაც ეწოდება საწყისი ხეების სიმრავლე.

(5) A არის სასრული ხეების სასრული სიმრავლე, რომელსაც ეწოდება დამხმარე ხეთა სიმრავლე.

ხის შიგა კვანძები მონიშნულია არატერმინალური სიმბოლოებით, რომელთაც აღვნიშნავთ დიდი ლათინური ასოებით, ხოლო ხის ფოთლები მონიშნულია ტერმინალური სიმბოლოებით ან არატერმინალური სიმბოლოებით. ტერმინალურ სიმბოლოებს აღვნიშნავთ პატარა ლათინური ასოებით. ხის ფოთოლი, რომელიც მონიშნულია არატერმინალური სიმბოლოთი მიწერილი აქვს სიმბოლო $\downarrow$, რომელიც აღნიშნავს, რომ ეს კვანი განკუთვნილია ჩასმისათვის. არსებობს მხოლოდ ერთი კვანძი, რომელიც მონიშნულია იგივე სიმბოლოთი, რაც ხის ძირი და ამ კვანს ეწოდება კვალი და იგი დამატებით მონიშნულია სიმბოლოთი * . ლექსიკალიზებულ TAG-ში ყველაზე მცირე ერთი მაინც ტერმინალური სიმბოლო (კვალი) უნდა იყოს საწყის ან დამხმარე ხეში. ხეს აღებულს $\{ I \cup A \}$ -დან ეწოდება ელემენტარული ხე. ასეთ ხეს ვუწოდებთ X ტიპის ელემენტალურ ხეს თუ მისი ძირი მონიშნულია X -ით.

ხე, რომელიც შედგენილია ორი ხის კომპოზიციით ეწოდება გამოყვანილი ხე. TAG იყენებს შეერთებისა და ჩასმის კომპოზიციურ ოპერაციებს. შეერთების ოპერაცია

β დამხმარე ხისა და ნებისმიერი α ხისაგან აგებს ახალ γ ხეს. ვთქვათ n არის α -ს არა ჩასმითი კვანძი მონიშნული X-ითა და β-ს ძირი მონიშნულია X -ით. γ, რომელიც მიიღება β-ს α-ს n-ურ კვანძთან შეერთებით და განისაზღვრება შემდეგნაირად:

1. n-ით განსაზღვრული ქვეხე, ვუწოდოთ მას δ , ჩამოეჭრება α -ს.

2. β - დამხმარე ხე მიუერთდება n კვან'ში α-ს და β-ს ძირი გაიგივდება n კვანთან.

3. β-ს კვალს მიუერთდება δ ქვეხე და δ ძირი გაიგივდება β-ს კვალთან.

ჩასმის ოპერაცია მოქმედებს მხოლოდ ხის ფოთლებზე, რომლებიც მონიშნული არიან არატერმინალური სიმბოლოებით და $\downarrow$-ით. კვანძი, სადაც ხდება ჩასმა შეიცვლება ჩასასმელი ხით. შესაძლებელია შემოვიტანოთ შეზღუდვები შეერთების ოპერაციაზე. მაგალითად, შესაძლებელია მოცემულ კვან'ში განვსაზღვროთ T დამხმარე ხეების სიმრავლე $T \subseteq A$, რომლის ელემენტი შეიძლება შევაერთოთ

მოცემულ კვან'ში. ასეთი შეზღუდვა აღვნიშნოთ SA(T)-თი. ასეთ შემთხვევაში შეერთება არაა სავალდებულო. NA-თი აღვნიშნოთ SA(T), სადაც T ცარიელი სიმრავლეა. ე. ი. მოცემულ კვანძში შეერთება აკრძალულია. OA(T)-თი აღვნიშნოთ აუცილებელი შეერთება, როცა მოცემულ კვანძში აუცილებლად უნდა მოხდეს შეერთება რომელიმე t∈T-თი. თუ არავითარი შეზღუდვები არაა შეერთებაზე და არა გვაქვს ჩასმის კვანძები ელემენტარულ ხეებში, ასეთ TAG-ს ეწოდება ხეთა შეერთების მკაცრი გრამატიკა. გამოყვანა გრამატიკაში განსაზღვრულია ისეთი ხის საშუალებით, რომელიც გვიჩვენებს თუ როგორ მიიღება საბოლოო ხე საწყისი ხისაგან შეერთებისა და ჩასმის ოპერაციების გამოყენებით; გრამატიკას ეწოდება ლექსიკალიზებული თუ იგი შედგება:

1. სტრუქტურათა სასრული სიმრავლისაგან, სადაც ყოველი სტრუქტურა დაკავშირებულია რაიმე ლექსიკურ ერთეულთან და ამ ლექსიკურ ერთეულს ეწოდება ამ სტრუქტურის კვალი.

2. რაიმე ოპერაციისაგან ან ოპერაციებისაგან სტრუქტურათა კომპოზიციისათვის მოითხოვება, რომ კვალი არ უნდა იყოს ცარიელი ლექსიკური ერთეული. ლექსიკონი შედგება სტრუქტურათა სასრული სიმრავლისაგან, სადაც ყოველი სტრუქტურა დაკავშირებულია რაიმე კვალთან. ლექსიკონით განსაზღვრულ სტრუქტურებს ეწოდებათ ელემენტარული სტრუქტურები, ხოლო სტრუქტურები რომლებიც მიიღებიან სტრუქტურათა კომბინირებით ეწოდებათ გამოყვანილი სტრუქტურები. მოითხოვება, რომ სტრუქტურა იყოს სასრული ზომის და ოპერაციას გადაჰყავდეს სტრუქტურათა სასრული სიმრავლე სასრული რაოდენობის მქონე სტრუქტურებში. სტრუქტურა არის ლექსიკალიზებული თუ არსებობს ერთი მაინც ლექსიკური ერთეული (არაცარიელი), რომელიც გვხვდება მასში. გრამატიკა, რომელიც შედგება ლექსიკალიზებული სტრუქტურებისაგან არის ლექსიკალიზებული. შესა'ლებელია კონტექსტისაგან თავისუფალი გრამატიკების ლექსიკალიზაცია TAG-ებით. გრამატიკული ფორმალიზმების ლექსიკალიზაცია საინტერესოა როგორც ლინგვისტური, ასევე ფორმალური თვალსაზრისით. თუ მივიღებთ თვალსაზრისს, რომ არ იყოს სრულად განცალკევებული თავიანთი ლექსიკური რეალიზაციისაგან, მაშინ ყოველი ელემენტარული სტრუქტურა სისტემურად დაკავშირებულია თავის ლექსიკურ კვალთან ლექსიკალიზებული მიდგომის დროს. ეს სტრუქტურები აზუსტებენ გაფართოებულ არეებს, რომლებზედაც შეი'ლება დავადოთ შეზღუდვები. კონტექსტისაგან თავისუფალი წესების ლექსიკალიზაციის პროცესი გვაი'ულებს ჩვენ გამოვიყენოთ ჩასმისა და შეერთების ოპერაციები სტრუქტურათა კომბინირებისათვის, ეს კი ხდის ფორმალიზმს ისეთს, რომელიც ხვდება კონტექსტის მგრ'ნობიარე ენათა კლასში. ჩასმისა და შეერთების ოპერაციები აიოლებენ კონტექსტისაგან თავისუფალი ენების ლექსიკალიზაციას. ასეთი ფორმალიზმი კი გვა'ლევს ლექსიკალიზებულ ხეთა შეერთების ენას. TAG, თავის მხრივ, ჩაკეტილია ლექსიკალიზაციის მიმართ.

სტეკიანი ავტომატები

ახლა განვიხილოთ უფრო სრულყოფილი ავტომატების მოდელი ვიდრე განხილული მაღაზიური ავტომატები იყო. ამ ავტომატების გამოთვლითი სიძლიერე აღწევს ტიურინგის მანქანის შესაძლებლობებს. ასეთი ავტომატი შეიძლება ეფექტურად გამოყენებულ იქნეს როგორც პროგრამირების რაიმე ენისათვის კომპილიატორის შესაქმნელად, ასევე ბუნებრივი ენების სინტაქსური ანალიზისათვის. სტეკიანი ავტომატი შეიცავს შესასვლელ ჯაჭვს, მდგომარეობებსა და სტეკს. მას შეუძლია წაიკითხოს სიმბოლო შესასვლელი ჯაჭვიდან ე. ი. მისმა მიმთითებელმა იმოძრაოს ჯაჭვის თავიდან მისი ბოლოს მიმართულებით ან პირიქით, შეუძლია სტეკის თავში ჩაწეროს რაიმე სიმბოლო ან წაშალოს, ასევე შესაძლებელია სტეკის მიმთითებლის მო'რაობა სტეკის შიგნით. თუ ასეთ ავტომატს მივცემთ შესაძლებლობას მოახდინოს ჩაწერა სტეკის შიგნითაც, მაშინ მისი შესაძლებლობები კიდევ უფრო გაიზრდება. სტეკური ავტომატი მუშაობს შემდეგნაირად:

1. იგი გადადის ერთი მდგომარეობიდან მეორეში.

2. მას შეუძლია გადაადგილდეს ერთი სიმბოლოთი შესასვლელი ჯაჭვიდან წაკითხული სიმბოლოს მარჯვნივ ან მარცხნივ.

3. მას შეუძლია შეასრულოს შემდეგი მოქმედებებიდან ერთ-ერთი.

a. გადაადგილოს სტეკის მიმთითებელი ერთი სიმბოლოთი სტეკიდან წაკითხული სიმბოლოს მარჯვნივ ან მარცხნივ.

b. თუ იგი კითხულობს სტეკის თავში მოთავსებულ სიმბოლოს, მაშინ მას შეუძლია ჩაწეროს სტეკის თავში სიმბოლოები და წაშალოს სტეკის თავიდან წაკითხული სიმბოლო. ფორმალურად $A = (K , \Sigma , \not\in ,\$,\Gamma , \delta , q_0,Z_0, F)$

სტეკიანი ავტომატი შედგება შემდეგი ელემენტებისაგან:

1. K არის მდგომარეობის სასრული არაცარიელი სიმრავლე;

2. Σ არის შესასვლელი ჯაჭვის სიმბოლოების სასრული არაცარიელი სიმრავლე;

3. $\notin$ და $ არიან სიმბოლოები, რომლებიც არ ეკუთვნიან Σ-ს და გვხვდებიან შესასვლელი ჯაჭვის თავსა და ბოლოში შესაბამისად , რომლებიც გამოიყენებიან შესასვლელი ჯაჭვის თავსა და ბოლოს დასაფიქსირებლად;

4. Γ არის სტეკის სიმბოლოების სასრული არაცარიელი სიმრავლე;

5. $Z_0 \in \Gamma$ და გვხვდება სტეკის ბოლოში მხოლოდ. იგი გამოიყენება იმის აღსანიშნავად, რომ მივაღწიეთ სტეკის ძირს;

6. δ არის ფუნქცია, რომლის განსაზღვრის

არეა K X (∑ ∪ { ⊄ ,$}) X Γ და მნიშვნელობათა არეა

{ – 1 , 0 , 1 } X K X { – 1 , 0 , 1 } X Γ * ისეთი რომ ყოველი q, a, და Z-ისათვის:

a. თუ δ (q, a, z)=(d,q′ , e,w) და w≠ z, მაშინ e = 0;

b. თუ δ (q,a,z$_0$)=(d, q′ , e,y), მაშინ y=z$_0$w,

w∈ Γ$_*$;

7. q$_0$∈ K და ეწოდება საწყისი მდგომარეობა;

8. F∈ K და ეწოდება საბოლოო მდგომარეობათა სიმრავლე. სტეკიანი ავტომატის შესასვლელ ჯაჭვი არის ∑ * $-ის ელემენტი. საბოლოო მდგომარეობები გამოიყენება იმისათვის, რომ გამოვიტანოთ ის ჯაჭვები, რომლებიც ამ გრამატიკით განსაზღვრულ ენას ეკუთვნიან. ჩანაწერი δ (a, q,z,)=(d,q′ ,e,z) ნიშნავს. რომ თუ სტეკიანი ავტომატი (SA) იმყოფება q მდგომარეობაში, სტეკის თავში არის Z და შესასვლელ სტრიქონიდან კითხულობს a სიმბოლოს მაშინ:

1. იგი გადადის q′ მდგომარეობაში;

2. იგი მოძრაობს მარცხნივ (შესასვლელი სტრიქონის მიმართ), თუ d=-1 ან მარჯვნივ თუ d=1 ან არ მოძრაობს თუ d=0;

3. იგი მოძრაობს (სტეკის მიმართ) მარცხნივ თუ e= -1 ან მარჯვნივ თუ e=1 ან არ მოძრაობს თუ e=0.

ჩანაწერი δ (q,a,z)=(d,q′ , o,w) და Z≠ W ნიშნავ რომ თუ SA არის (q,a,z) კონფიგურაციაში, მაშინ

1. იგი მოძრაობს შესასვლელ სტრიქონზე d-ს მიხედვით;

2. გადადის q′ მდგომარეობაში;

3. სტეკში წერს W-ს Z-ის ნაცვლად.

6a პირობა კრძალავს სტეკზე მოქმედებებს, როცა e=0 და 6b პირობა კრძალავს მაღაზიის დაცარიელებას, ე.ი. Z_0-ის წაშლას. სტეკიანი ავტომატის მიმდინარე მდგომარეობა განისაზღვრება წასაკითხავი სტრიქონით, მიმდინარე q მდგომარეობითა და სტეკის შიგთავსით, ე.ი. SA ავტომატის მდგომარეობა (M) არის

K X Q(∑ ∪ { e,$,a}) * X (Γ ∪ { b}) * სიმრავლის ელემენტი, სადაც a არის შესავალი სტრიქონის მიმდინარე სიმბოლოს მიმთითებელი, ხოლო b არის სტეკის თავის მიმთითებელი. ახლა განვსაზღვროთ | — დამოკიდებულება SA ავტომატისათვის. ვთქვათ SA= (K , ∑ , ∉ $, Γ ,δ , q_0, Z_0,F), | — დამოკიდებულება SA –ს მდგომარეობებს შორის განისაზღვრება შემდეგნაირად: თუ k,l≥ 1, a_1,a_2 , ..., a_k ეკუთვნის ∑ ∪ { ∉ $} ; a_{k+1}| —, Z_1,...,Z_e ეკუთვნის Γ , Y ეკუთვნის Γ * ეკუთვნის Γ -ს, მაშინ

1. თუ δ (q, a_i,Z_j)=(d,q′ ,e,z_j), სადაც 1≤ i ≤ k და 1≤ j≤ l აკმაყოფილებს პირობებს: i. d≥ 0 თუ i=1; ii. e≥ 0 თუ j=1და iii. e≤ 0 თუ j=l, მაშინ (q,a_1...āa_i...a_k, z_1...Z_jb...Z_l)| — (q′ ,a_1...āa_{i+d} a_{k+1}, z_1 ...z_{j+e}b...z_l);

2. თუ δ (q, a_i, Z)= (d,q′ ,0,w) და 1≤ i≤ k აკმაყოფილებს i პირობას, მაშინ (q,a_1...āa_i...a_k, yzb)| — (q′ , a_1...ba_{i+d}...a_k, ywb).

ამ დამოკიდებულებით ჩვენ შეგვიძლია აღვწეროთ SA ავტომატის ერთი ნაბიჯი. ახლა განვიხილოთ ამ დამოკიდებულების ტრანზიტული და რეფლექსური ჩაკეტვა * :

(q, xāy, w_1bw_2) | — * (q ′ ,x′ āy′ ,w′ ,bw_2),′ სადაც X,Y,X′ და y′ ჯაჭვებია (∑ ∪ { ∉ $}) *-დან და W_1,W_2,W_1′ ,′W_2′ ჯაჭვებია Γ *-დან. თუ K ≥ 0 , f_i, g_i და h_i, სადაც 0 ≤ i≤ k ისეთი, რომ f_0=q, f_k=q′ , g_0=xāy; g_k=x′ by′ , h_0=w_1bw_2,h_k=w′ $_1$bw′ $_2$, მაშინ

(f_i,g_i,h_i) | — (f_{i+1}, g_{i+1},h_{i+1}), სადაც 0 ≤ i ≤ k.

ვიტყვით, რომ SA ავტომატი უშვებს X∈ ∑ * ჯაჭვს თუ

(q_0, ā∈ x$,$z_0$b) | — *(q,∉ x$ā,w,bw_2),

სადაც q∈ F და W_1,W_2∈ Γ *. SA ავტომატის მიერ დაშვებული ყველა ჯაჭვთა სიმრავლე აღვნიშნოთ L(SA)-თი და იგი არის ენა განსაზღვრული SA ავტომატით. მაღაზიური ავტომატის ანალოგიურად, აქაც ჩვენ შეგვი'ლია შემოვიტანოთ არადეტერმინისტული SA ავტომატის განმარტება (იხ. Ginsburg...). SA ავტომატზე გარკვეული შეზღუდვების დადებით მიიღება ყველა აქამდე განხილული

ავტომატები. დამოკიდებულება სხვადასხვა ავტომატებსა და ცნობილ სიმრავლეებს შორის მოკლედ შეი'ლება წარმოვადგინოთ შემდეგი დიაგრამით

აკლია დიაგრამა

ამ დიაგრამაზე გამოყენებულია შემდეგი აღნიშვნები: 1 = ერთმიმართულებიანი, 2= ორმიმართულებიანი, D=დეტერმინისტული, FA=სასრული, დეტერმინისტული ავტომატის, N=არადეტერმინისტული, NE=არაწაშლადი, RS=რეკურსიული სიმრავლეები, P= მაღაზიური ავტომატი, L_{BA}= წრფივადდაკავშირებული ავტომატი, S=სტეკური ავტომატი, RF=რეკურსიულად გადათვლადი სიმრავლეები. ორმაგი ხაზები გვიჩვენებენ საკუთრივ ნაწილს, ერთმაგი ხაზი გვიჩვენებს რომ იგი წარმოადგენს ნაწილს და საკუთრივ ნაწილს წარმოადგენს თუ არა არაა ცნობილი. მაგალითად, a═b ნიშნავს რომ b⊂ a, a— b ნიშნავს b⊆ a. ცნობილია ,რომ SA ავტომატის მიერ დაშვებული სიმრავლეები არიან რეკურსიული სიმრავლეები.

შემოვიტანოთ სტეკიანი სიმრავლეების განმარტება: რაიმე სტეკიანი სისტემა არის სამეული G=(K,Γ , P), სადაც K და Γ არიან სასრული არაცარიელი სიმრავლეები $K \cap \Gamma = = \varnothing$ და P არის შემდეგი სახის წესების სასრული სიმრავლეები (სადაც S და S′ ეკუთვნის K-ს და A და B ეკუთვნის Γ -ს):

1. ჩაწერა: $QAs \rightarrow QABs'$

2. შეცვლა: $QAs \rightarrow QBs'$

3. წაშლა: $QAs \rightarrow Qs'$

4. დატოვება: $Q_1AsQ_2 \rightarrow Q_1AsQ_2$

5. მარცხნივ გადაწევა: $Q_1AsQ_2 \rightarrow Q_1s'\ AQ_2$

6. მარჯვნივ გადაწევა: $Q_1AsBQ_2 \rightarrow Q_1ABs'\ Q_2$

ახლა განვსაზღვროთ თუ როგორ უნდა გამოვიყენოთ ზემოთა წესები. $Q,Q_1,Q_2 \in (K \cup \Gamma)^*$. ვთქვათ G=(K,Γ , P) არის სტეკიანი სისტემა. თუ W,Y,Y′ ეკუთვნის $(K \cup \Gamma)^*$

და $Q_1YQ_2 \rightarrow Q_1Y' Q_2$, მაშინ ვწერთ $WY \Rightarrow WY'$. ანალოგიურად, თუ W,W′ ,Y,Y′ ეკუთვნის ($K \cup \Gamma$) * და $Q_1YQ_2 \rightarrow Q_1Y' Q_2$, მაშინ ვწერთ $WYW' \Rightarrow WY' W'$. თუ X და Y ეკუთვნის ($K \cup \Gamma$) * , მაშინ $X \Rightarrow * Y$ ნიშნავს, რომ არსებობს Z_0,...Zr ხაჭვები ($K \cup \Gamma$) * -დან ისეთი რომ Z_0

=X, Zr=Y, Zi⇒ Zi+1 და $0 \leq i \leq r$. დამოკიდებულება ⇒ არის რეფლექსური და ტრანზიტული. ასეთი სისტემის გამოყენებით მტკიცდება წინა დიაგრამაზე ნაჩვენები მთელი რიგი დებულებებისა (იხ.). SA ავტომატით დაშვებული სიმრავლეები არიან ჩაკეტილი თანაკვეთის ოპერაციის მიმართ.

CAT2 ენის აღწერა

CAT 2 არის გამიზნული ბუნებრივი ენის კომპიუტერული დამუშავებისათვის. იგი შეიცავს ისეთ საშუალებებს რომელთა დახმარებით გაადვილებულია ბუნებრივი ენის სტრუქტურების კომპიუტერზე წარმოდგენა. პირველ რიგში იგი გამიზნულია მანქანური თარგმანისათვის, თუმცა მისი საშუალებით შესაძლებელია ბუნებრივი-ენის ანალიზი, ცოდნის დაგროვება და მისი გამოყენება სხვადასხვა მიზნისათვის. CAT2-ს საფუძვლად აქვს მიდგომა, რომელიც შემუშავებული იყო EUROTRA პროექტში კერძოდ < C,A > , T-ს სახელით ცნობილი მიდგომა მანქანური თარგმანისადმი (Arnold და სხვები 1985, 1986, Arnold და Tombe 1987). იგი დაფუ'ნებულია კონსტრუქტორებისა, ატომებისა და ტრანსლიატორების ცნებებზე. კონსტრუქტორები და ატომები ქმნიან წარმოდგენის რაიმე დონეს, ხოლო ტრანსლიატორები აკავშირებენ ერთიმეორესთან რაიმე ორ დონეს. ენობრივი სტრუქტურები წარმოიდგინებიან ხეების საშუალებით. მას აქვს გამარტივებული სინტაქსი და სემანტიკა. კარგად განსაზღვრულია, რომელიც ემყარება თვისებათა ლოგიკასა (Johnson 1988) და თვისებათა უნიფიკაციის ფორმალიზმს (Kasper 1987). CAT2-'ირითადი განსხვავება სხვა მსგავს ფორმალიზმებისაგან არის ის , რომ იგი იყენებს ლინგვისტურ-სტრუქტურების წარმოსადგენადაც ხეებს, მაშინ როდესაც თვისებათა სტრუქტურები გამოიყენება მაგალითად: LFG-ში (Kaplan და Bresnan 1982), FUG-ში (Kay 1984) და HPSG-ში (Pollard and Sag 1987). CAT2-ის სხვა დამახასიათებელი ნიშანია დონეთა როლები, რომლებიც ახასიათებენ წარმოდგენის დონეებს. ამგვარად, KAT2-ში ჩვენ გვაქვს სხვადასხვა წარმოდგენის დონეები, რომლებიც დაკავშირებული არიან ერთიმეორესთან ტრანსლიატორებით. წარმოდგენის დონე განსაზღვრულია გენერატორით.

ძირითადი ცნებები

რაიმე ბუნებრივი ენის გრამატიკა CAT2-ში წარმოიდგინება როგორც წესების სიმრავლე, რომელიც დაჯგუფებულია გენერატორებად და ტრანსლიატორად. რაიმე გენერატორი არის წესების სიმრავლე, რომელიც განსაზღვრავს წარმოდგენის დონეს და რაიმე ტრანსლიატორი-არის წესების სიმრავლე, რომელიც აკავშირებს

ერთიმეორესთან წარმოდგენის ორ დონეს. ერთი გენერატორი განსაზღვრავს სინტაქსურ სტრუქტურებს, რომელსაც ეწოდება სინტაქსური გენერატორი და მეორე გენერატორი განსაზღვრავს რეალურ სტრუქტურებს, რომელსაც ეწოდება რეალური გენერატორი. ასევე არსებობენ მორფოლოგიური გენერატორი და ტექსტის გენერატორი. მორფოლოგიური გენერატორი განსაზღვრავს მორფოლოგიურ სტრუქტურებს და ტექსტის გენერატორი გათვალისწინებულია მრავალ-წინადადებიანი სტრუქტურებისათვის. ამგვარად ავაგებთ რა სინტაქსურ გენერატორს და ერთს ან მეტ რეალურ გენერატორებს, რომელთაც დააკავშირებენ ტრანსლიატორები, შესაძლებელია მოცემული ენისათვის შეიქმნას ლინგვისტური სტრუქტურები კომპიუტერის საშუალებით. ასევე მეორე ენისათვის თუ გავაკეთებთ იგივეს და მათ გენერატორებს დავაკავშირებთ ტრანსლიატორებით ამით შესრულდება ერთი ენის ლინგვისტური სტრუქტურების გადაყვანა მეორე ენის შესაბამის ლინგვისტურ სტრუქტურებში. მაგალითად, განვსაზღვროთ ინგლისური ენის გრამატიკა, რომელიც შედგება CSEN სინტაქსური გენერატორისაგან და ISEN ინტერფეისული გენერატორისაგან და ისინი დავაკავშიროთ ტრანსლიატორებით: CSEN→ ISEN, რომელსაც გადაჰყავს სინტაქსური სტრუქტურები CSEN-იდან რეალურ სტრუქტურებში ISEN-ში და ISEN→CSEN, რომელიც ასრულებს შებრუნებულ მოქმედებას. ასევე განვსაზღვროთ ქართული ენის გრამატიკა CSKR და ISKAR გენერატორებით და ტრანსლიატორებით CSKAR→ ISKAR და ISKAR→ CSKAR. დავუმატოთ მათ ტრანსლიატორები: ISEN→ ISKAR და ISKAR→ ISEN, მაშინ შეგვი'ლია ჩვენ ქართული IS (ინტერფეისული სტრუქტურები) გადავიყვანოთ ინგლისურ IS სტრუქტურებში და პირიქით. ლინგვისტურ სტრუქტურებს, რომელსაც გენერატორი ჰქმნის ეწოდებათ ობიექტები. კონკრეტულ ობიექტს აქვს ხის ფორმა, რომელშიც კვანძები შეიცავენ თვისებების სიმრავლეს. ყოველი თვისება არის წყვილი ატრიბუტი და მნიშვნელობა. ყოველი ატრიბუტი არის უნიკალური ხის კვან'ში და ყოველ ატრიბუტს აქვს მნიშვნელობა— ატომური კონსტანტა ან ცვლადი ან თვისებათა სიმრავლე. ობიექტების გენერირებისათვის CAT2 იყენებს უნიფიკაციის ოპერაციას, როგორც ხეების ასაგებად, ასევე თვისებათა შესადგენად. უნიფიკაცია რაიმე ობიექტისა ხდება მარცხნიდან მარჯვნივ ქვეხეების გათვალისწინებით. უნიფიკაციასთან დაკავშირებული მნიშვნელოვანი ცნებაა შეზღუდვის დაკმაყოფილება. შეზღუდვები, რომლებიც არ კმაყოფილდება მოცემულ მომენტში, ხდება მისი გადადება მანამდე, სანამ ახალი ინფორმაციის მიღება არის შესაძლებელი. შეზღუდვათა ტიპებია: დადებითი, უარყოფითი, დიზიუნქციური, არსებობის და პირობითი შეზღუდვები.

თვისებები და შეზღუდვები

თვისებები გამოიყენება იმისათვის, რომ მივუთითოთ ლინგვისტური სტრუქტურის კონკრეტული მახასიათებლები. თვისება შედგება წყვილისაგან — ატრიბუტი და მნიშვნელობა, რომლებიც ერთიმეორისაგან გამოიყოფა ′ = ′ ნიშნით. თვისებათა

სიმრავლეს ათავსებენ ფიგურულ ფრჩხილებში და ერთიმეორისაგან გამოიყოფიან მ'იმეებით. მაგალითად:

{ cat=არს, lex=სახლები, lu=სახლი, arg={ per=3, num=მრ} , wh=′ – ′ }

თვისების მნიშვნელობა შეი'ლება იყოს კონსტანტა, ცვლადი ან თვისებათა სიმრავლე. თვისება შეი'ლება იყოს რთული (მაგ. arg) ან ატომური (მაგ.cat, lu და სხვა). CAT2-ის კონსტანტებია PROLOG-ის კონსტანტები, მხოლოდ არ გამოიყენება ნამდვილი რიცხვები. თვისების მნიშვნელობა შეიძლება იყოს ცვლადი. ცვლადი შეიძლება დაუკავშირდეს კონსტანტას ან თვისებათა სიმრავლეს უნიფიკაციით და შემდეგ ხდება ინიციალიზებული ცვლადი. ასევე ცვლადი შეიძლება დაუკავშირდეს სხვა ცვლადს ასეთ შემთხვევაში გვაქვს დაკავშირებული ცვლადი. თვისებათა სიმრავლის აღწერა გამოსახავს შეზღუდვებს ობიექტის თვისებებზე.

დადებითი შეზღუდვები ისეთი შეზღუდვებია, რომლებიც მოიცემა კონსტანტებით ან ცვლადებით ან შედგენილი თვისებით, რომელიც კვლავ შედგება დადებითი შეზღუდვებით. უარყოფითი შეზღუდვები შეიცავენ უარყოფებს. მაგ . {per¬ =3}. დიზიუნქტური შეზღუდვები აცხადებენ, რომ ობიექტის თვისებების მნიშვნელობა შეიძლება იყოს ერთ-ერთი რამოდენიმე ალტერნატივებიდან. არსებობის შეზღუდვები მოითხოვენ, რომ მოცემული შეზღუდვა უნდა არსებობდეს ობიექტში წინააღმდეგ შემთხვევაში იგი განიცდის მარცხს. არსებობის შეზღუდვა აღინიშნება = = ნიშნით. პირობით შეზღუდვები არიან შეზღუდვები რაიმე თვისებათა სიმრავლეზე და არა რაიმე თვისების მნიშვნელობაზე. პირობით შეზღუდვას აქვს სახე:

პირობა > > მიმდევრობა,

სადაც როგორც პირობა ასევე მიმდევრობა არიან თვისებათა სიმრავლის აღწერები. თუ ობიექტის თვისებათა სიმრავლე აკმაყოფილებს შეზღუდვებს მოცემულს პირობაში, მაშინ იგი უნდა აკმაყოფილებდეს შეზღუდვებს მოცემულს მიმდევრობაში.

არსებობს ორი თვისების აღწერის მნიშვნელობები CAT2-ში, რომელთაც აქვთ სპეციალური დანიშნულება. ერთი არის INDEX-უნიკალური მთელი რიცხვების გენერირებისათვის და მეორე არის NUM-ისათვის, რომ მნიშვნელობა იყოს რიცხვითი. ისინი შეიძლება შეგვხვდეს როგორც მნიშვნელობები ნებისმიერი თვისების აღწერაში. ორივენი წარმოადგენენ შეზღუდვებს მნიშვნელობებზე. $INDEX მნიშვნელობა გამოიყენება ანაფორულ მითითებაში ინდექსების გენერირებისათვის.

გენერატორები

რაიმე გენერატორი განსაზღვრავს ობიექტების სიმრავლეს წარმოდგენის რაიმე დონეზე. გენერატორები შეიცავენ ორი ტიპის წესებს: b-წესებსა და f-წესებს.

b-წესები (აგების წესები) არიან კონსტრუქტორთა სიმრავლე, რომელიც განსაზღვრავს სტრუქტურულ ხეებს და f-წესები (თვისებათა წესები) მოქმედებენ თვისებებზე ხის შიგნით, ან ამატებენ ახალ თვისებებს ან ამოწმებენ უკვე არსებულ თვისებებს. როცა ჩვენ ვქმნით რაიმე ფაილს, რომელიც შეიცავს რაიმე გენერატორის განსაზღვრას, ჩვენ ვიწყებთ განსაზღვრებას კომპილატორის დირექტივით @ level:

@ level (NAME /TYPE / ATTRIBUTE).

ეს დირექტივა ა'ლევს გენერატორს სახელს, (NAME) განსაზღვრავს მის ტიპს (TYPE) და პრივილიგირებულ ატრიბუტებს მისთვის. (ATTRIBUTE) სახელი შეი'ლება იყოს, ატომური კონსტანტა. ტიპი არის სინტაქსური (Syntactic) ან რელაციური (relational) და ატრიბუტი არის რაიმე ატომური კონსტანტა. მაგალითად

@ level (CSKAR / syntactic / cat).

მომდევნო ინსტრუქციები ახალ @ level ინსტრუქციამდე ეკუთვნიან ამ გენერატორს.

B-წესები განსაზღვრავენ კონსტრუქტორებსა და ატომებს, რომლებიც აღწერენ ცალკეულ ხეებს. ყოველ b-წესს აქვს ფორმა:

NAME=ROOT.BODY.

სადაც სახელი (NAME) არის ატომური კონსტანტა, რომელიც აღნიშნავს წესს, ძირი (ROOT) არის რაიმე თვისებათა სიმრავლის აღწერა, რომელიც აღწერს ხის ძირს და ტანი (BODY) არის უშუალო შთამომავლების თვისებათა სიმრავლეების აღწერები. ატომურიკონსტრუქტორის შემთხვევაში ტანი არის ცარიელი სია. b-წესები იწყება დირექტივით

@ rule (b), ამ დირექტივის მომდევნო ინსტრუქციები შემდეგ @ rule ან @ level დირექტივებამდე განიხილებიან როგორც b-წესები.

საუბრის წარმოდგენის თეორია

საუბრის წარმოდგენის თეორია (DRT) დაკავშირებულია ბუნებრივი ენის ტექსტის სემანტიკური სტრუქტურის წარმოდგენასთან. უფრო ზუსტად, იგი ცდილობს ტექსტის შემადგენელი მიმდევრობითი წინადადებებისათვის წარმოგვიდგინოს მათი სინტაქსური, სემანტიკური და ლოგიკური სტრუქტურა და ის კავშირები, რომლებითაც ისინი დაკავშირებული არიან ერთიმეორესთან. ამასთანავე ეს წარმოდგენა უნდა იყოს განზოგადებული ე. ი. ისეთი, რომელიც გამოდგებოდა არა მარტო მოცემული ტექსტის წინადადებებისათვის არამედ აგრეთვე ბევრი სხვა გარკვეული თვალსაზრისით მსგავსი წინადადებებისათვის. DRT არის ბუნებრივი ენის სემანტიკის მეთოდი და წარმოიშვა თავსატეხი წინადადებების (მათი სემანტიკის თვალსაზრისით) სემანტიკის დასადგენად და აგრეთვე დროის და ასპექტის გარჩევისათვის რომანულ ენებში. ამ მიდგომას საფუ'ვლად უდევს მოდალური თეორიული ს ემანტიკა, რომელიც შემოტანილია მონტეგიუს [ ] მიერ და წარმატებით იყო გამოყენებული მის მიერ ინგლისური ენის ფრაგმენტისათვის. დღეისათვის იგი ცნობილია მონტეგიუს გრამატიკის სახელწოდებით. მონტეგიუს გრამატიკამ განვითარება ჰპოვა პარტიეს (Partee [ ]), თომასონის (Thomasson [ ]) , ფოდორის (Fodor [ ]), კეენანის (Keenan [ ]) და იანსენის (Iansen [ ]) შრომებში. DRT არ არის წმინდა მოდელურ თეორიული სემანტიკის მეთოდი. DRT ცდილობს გაასწოროს მოდელურ თეორიული პარადიგმების ცალმხრივობა რეფერენციალური პერსპექტივის ინტერპრეტაციაზე ორიენტირებულ თვალსაზრისთან კომბინირებით. იგი არის ლინგვისტური მნიშვნელობის უფრო რთული მეთოდი, რადგანაც იგი ერთის მხრივ საგანზე ორიენტირებულია და მეორეს მხრივ იგი დაკავშირებულია ენის ინტერპრეტაციასთან. ეს თეორია განვითარებულია კამფის (Kamp [ ]) და ჰეიმის (Heim [ ]) მიერ. ჩვენ შევეცდებით აღვწეროთ ეს თეორია და სათანადო საუბრის წარმოდგენის სტრუქტურები (DRS) და მათი აგების წესები ქართული ენის ფრაგმენტის მაგალითზე. სანამ ამ თეორიის აღწერას შევუდგებოდეთ გავეცნოთ ზოგიერთ ძირითად ცნებებს, რომლებიც ჩვენ დაგვჭირდება DRT-ს აღწერისას. როდესაც, ჩვენ ვკითხულობთ ტექსტს და ვიგებთ მის შინაარსს ეს პროცესი შეიძლება გაგებულ იქნეს როგორც ლინგვისტური ფორმიდან შინაარსის აღების პროცესი და ამ შინაარსის ჩადება ახალ აზროვნებით ფორმაში. პირიქით, როცა ჩვენ გადმოვცემთ შინაარსს ტექსტის გენერირებისას, მაშინ ხდება შინაარსის აზროვნებითი ფორმიდან გადატანა (ჩადება) ლინგვისტურ ფორმაში.

ამგვარად, ეს პროცესები ჩვენ შეგვიძლია განვიხილოთ როგორც თარგმანი ერთი ენიდან მეორეზე (ბუნებრივი ენიდან აზროვნების ენაზე თარგმანი ან პირიქით). მეორეს მხრივ, ისმის საკითხი შინაარსის ჭეშმარიტების შესახებ ე.ი. ბუნებრივი ენის გამოსახულების (ან შესატყვისი აზროვნების ენის გამოსახულების), რომელსაც შინაარსი გააჩნია, ჭეშმარიტების შესახებ. სანამ განვმარტავდეთ თუ რას ნიშნავს ეს, მოვიყვანოთ ფოდორის (Fodor [ ]) არგუმენტები.

სინტაქსი

ცალსახა ენის მაგალითია წინადადებათა აღრიცხვის ენა U^0, სადაც ინდივიდუალური კონსტანტებია `a`, `b` და `c`. ერთადგილიანი პრედიკატებია (ანუ ერთადგილიანი ზმნები ანუ გარდაუვალი ზმნები) `p` და `q` ერთადგილიანი დამაკავშირებელია `¬` და ორადგილიანი დამაკავშირებლები `∧` და `∨`. ფორმულების სიმრავლე განსაზღვრულია რეკურსიულად, როგორც სემანტიკური სიმრავლე K ისეთი, რომ

(0) თუ არის ერთადგილიანი ზმნა და α არის რაიმე ინდივიდუალური კონსტანტა, მაშინ ⌜(α)⌝ ∈ K

(1) თუ ζ ერთადგილიანი დამაკავშირებელია φ ∈ K , მაშინ ⌜ζφ ⌝∈ K და

(2) თუ ς ორადგილიანი დამაკავშირებელია და φ , Ψ ∈ K , მაშინ ⌜ φ ζ Ψ ⌝∈ K . `p(a)`, `⌜ ⌝ p(a) ` , ` p(a) ∨ Q(a) და ` ⌝⌝ p(a) ∧ ⌝ Q(b) ფორმულებია. ამ ფორმულირებაში მონაწილეობს ხუთი სინტაქსური კატეგორია: ინდივიდუალური კოტსტანტები, ერთადგილიანი ზმნები, ერთადგილიანი დამაკავშირებლები, ორადგილიანი დამაკავშირებლები და ფორმულები. გარდა უკანასკნელისა, ყველა კატეგორია განსაზღვრულია მათი ელემენტების ჩამოთვლით. ფრჩხილების სამი წყვილი არცერთ კატეგორიას არ ეკუთვნიან. ისინი შემოტანილი არიან სინკატეგორიმატიკულად ფორმულათა სიმრავლის რეკურსიულად განსაზღვრისას. რაც იმას ნიშნავს, რომ ამ ენის სტანდარტული სემანტიკური ინტერპრეტაციის დროს მათ არავითარი დამოუკიდებელი მნიშვნელობა არა აქვთ. ფრჩხილების სემანტიკური გამოყენებაა არა მნიშვნელობათა შექმნა, არამედ ფორმულათა სინტაქსური ცალსახობის უზრუნველყოფა, ურომლისოდაც ენა იქნებოდა ომონიმური. ამის საილუსტრაციოდ განვიხილოთ L ენა, სადაც (2) შეცვლილია შემდეგი წესით: (2_1) თუ ζ არის ორადგილიანი დამაკავშირებელი და φ , Ψ ∈ K მაშინ φ ζΨ ∈ K . L ენა არის ომონიმიური, რადგან ფორმულას

`p(a)∨Q(a)∧Q(b)`აქვს ორი გარჩევა და შესაბამისად ორი განსხვავებული სინტაქსური სტრუქტურა:

(S1)აკლია

(S2)

სადაც S-ით აღნიშნულია განსახილველი ფორმულა. თუ a არის პეტრე, b არის თამარი, P არის დადის და Q არის ზის, მაშინ (S1) ნიშნავს ,რომ (პეტრე დადის) ან (პეტრე ზის და თამარი ზის), ხოლო (S2) ნიშნავს, რომ (პეტრე დადის ან პეტრე ზის) და (თამარი ზის). მრგვალი ფრჩხილები აღნიშნავენ `ან`-ისა და `და`-ს საზღვრებს. (S1) და (S2) არიან ანალიზის ხეები, რომლებიც გვიჩვენებენ თუ როგორ წარმოიქმნება წესების გამოყენებით S ფორმულა. ანალიზის ხეები გვიჩვენებენ ფორმულათა სინტაქსურ სტრუქტურებს, რომლებიც თავის მხრივ მიიღებიან რეკურსიული განსაზღვრებებით. სინტაქსური კატეგორიები, რომლებიც ფიგურირებენ ამ რეკურსიებში ერთ-ერთ მნიშვნელოვან თვისებებზე მიუთითებენ. თუ რაიმე ფორმულები ეკუთვნიან ერთდაიმავე სინტაქსურ კატეგორიას, მაშინ ისინი ერთნაირად შეი'ლება დაუკავშირდნენ სხვა გამოსახულებებს ერთიდაიგივე სორტის დაკავშირებით. ეს არის გამოყენებული ენის ფორმულების რეკურსიული განსაზღვრისას. დაკავშირების სორტები არიან ფუნქციები, რომლებიც იღებენ გამოსახულებათა მიმდევრობას და გვაძლევენ გამოსახულებებს. მაგალითის სახით განვსაზღვროთ ასეთი ფუნქციები U^0-ისთვის. ვთქვათ $F_0{}^0$ იყოს სორტი, რომელიც აკავშირებს ერთადგილიან ზმნას ინდივიდუალურ კონსტანტასთან და ღებულობს ფორმულას, $F_1{}^0$ იყოს სორტი, რომელიც იღებს ერთადგილიან დამაკავშირებელს და აკავშირებს მას რაიმე ფორმულასთან, რომ მიიღოს ახალი ფორმულა. $F_2{}^0$ იყოს სორტი, რომელიც აკავშირებს ორადგილიან დამაკავშირებელს ორ-ადგილიან ფორმულათა მიმდევრობასთან და ღებულობს ახალ ფორმულას. ამგვარად, გვაქვს განტოლებები, რომლებიც განსაზღვრავენ ამ ფუნქციებს:

$F_0{}^0(\delta, \alpha) = \ulcorner \delta(\alpha) \urcorner$

$F_1{}^0(\zeta,\varphi) = \ulcorner \zeta(\varphi) \urcorner$

ეს ფუნქციები ფაქტიურად არიან ოპერაციები, რომლებიც განსაზღვრულია ენის გამოსახულებებზე. $F_1{}^0(\ulcorner QV`,`b` \urcorner) = \ulcorner (QV(\,)b\,) \urcorner$ ამ ფუნქციებს ეწოდებათ U^0-ის სტრუქტურული ოპერაციები და გამოიყენებიან სინტაქსური წესების გამოსაცხადებლად, მაგრამ ისინი თვით არ არიან სინტაქსური წესები. ასე მაგალითად, ასებობს ისეთი წესი, რომელიც გვეუბნება, რომ $F_0{}^0$ არის სინტაქსური ოპერაცია და გამოყენებულია ერთადგილიანი ზმნისა და ინდივიდუალური კონსტანტისაგან ფორმულის მისაღებად. იმისათვის, რომ ასეთი წესი ჩამოვაყალიბოთ, ჩვენ პირველ რიგში უნდა ჩამოვთვალოთ U^0-ის სინტაქსური კატეგორიები: ICST, IV,1C, 2C და FOR. ვთქვათ ∇ იყოს სიმრავლე

{ICST,IV,1C,2C,FOR} . მონტეგიუს მიხედვით, ეს სიმრავლე არის ცალსახა ენის განსაზღვრების ერთ-ერთი კომპონენტთაგანი. სინტაქსური წესი უნდა შეიცავდეს სამი სახის ინფორმაციას: სტრუქტურულ F ოპერაციას, გამოსახულებათა სინტაქსურ კატეგორიებს, რომლებიც არიან F-ის არგუმენტები და F-ის მნიშვნელობის სინტაქსურ კატეგორიას. მაგალითად U^0-ის პირველი წესი ამბობს, რომ თუ ε არის IV და α არის ICST მაშინ $\varepsilon(\alpha)$ არის FOR. ამგვარად, მონტეგიუს სინტაქსური წესები არიან დალაგებული სამეულები, რომლის პირველი კომპონენტი არის სტრუქტურული ოპერაცია, მეორე კომპონენტია კატეგორიული ინდექსების

მიმდევრობა და მესამე ელემენტია კატეგორიული ინდექსი. ამგვარად S^0 სიმრავლის შემდეგი სამი სამეული შეადგენს U^0-ის სინტაქსური წესების სიმრავლეს:

$< F_0{}^{0,<}$IV,ICST> , FOR>

$< F_1{}^0$,< 1C, FOR> , FOR>

$< F_2{}^0$,< 2C, FOR, FOR > ,FOR>

სემანტიკური მოსაზრებით მიზანშეწონილია დავაფიქსიროთ ის კატეგორია, რომელიც შეადგენს ენის წინადადებებს, რადგან წინადადებებს უნდა ჰქონდეს ჭეშმარიტული მნიშვნელობები. U^0-ის შემთხვევაში ეს არის FOR. სიმრავლეები Δ^0 და S^0 ითვალისწინებენ U^0-ის სტრუქტურულ დახასიათებას. და მთელ რიგ კატეგორიების გამოსახულებებს აქვთ თავისი სტრუქტურა, გარდა იმ გამოსახულებებისა, რომლებიც შეადგენენ რეკურსიის ბაზისს. ასეთი გამოსახულებები არიან δ კატეგორიის გამოსახულებები და ისინი ეკუთვნიან ლექსიკონს, სადაც $\delta \in \Delta$. ლექსიკონი შეიცავს სიმრავლეს X δ , სადაც $\delta \in \Delta$. U^0-სთვის ასეთი სიმრავლეებია:

X^0_{ICST}={ `a`,`b`,`c`}

X_{IV}={ `p`,`a`}

X^0_{IC}={ ` ¬ `}

X^0_{2C}={ `V`,`∧ `}

X^0_{FOR}= ∧

X^0_{FOR}=∧ გვიჩვენებს, რომ ფორმულები არიან სინტაქსურად რთული. ვთქვათ, $C^{0\delta}$ იყოს U^0-ის δ კატეგორიის გამოსახულებათა სიმრავლე

C^0=< $C^{0\delta > \delta \in \Delta 0}$

C^0 შეიძლება განისაზღვროს S^0-ით, სინტაქსური წესებით და ოპერაციებით $F_i{}^0$ და Xδ ^{0}-ით

i∈ { 0, 1,2 } და $\delta \in \Delta^0$. ამგვარად, C^0-ის ინდექსირებული ოჯახი არის ისეთი უმცირესი ოჯახი C Δ^0-ით ინდექსირებული სიმრავლეებისა, რომ

(1) $X^{0\delta} \subseteq C\delta_{\ \delta \in \Delta}{}^0$;

(2.0) $\varsigma \in C_{IV}$ და $\alpha \in C_{ICST}$, $F_0{}^0(\varsigma , \alpha) \in C_{FOR;}$

(2.1) $\varsigma \in C_{1C}$ და $\varphi \in C_{FOR,}$ $F_1^0(\varsigma , \varphi) \in C_{FOR;}$

(2.2) $\varsigma \in C_{2C}$ და $\varphi \ {}_1\Psi \in C_{FOR,}$ $F_2^{0(\varsigma , \varphi , \Psi)} \in C_{FOR}$

განვიხილოთ U^0-ის სწორი გამოსახულების A_0 სიმრავლე იმ გამოსახულებებთან ერთად, რომლებიც მიიღებიან U^0-ის სტრუქტურული ოპერაციების გამოყენებით. იგი შეიცავს მთელ რიგ გამოსახულებებს, რომლებიც ნაკლებად საინტერესოა სინტაქსური თვალსაზრისით, მაგრამ საინტერესოა U^0-ის ზოგადი თვისებების შესწავლის თვალსაზრისით. ეს სიმრავლე შეგვიძლია განვიხილოთ როგორც ალგებრა U^0-ის სტრუქტურული ოპერაციებით. ამ ალგებრის საშუალებით შეგვიძლია შემოვიტანოთ ისეთი მათემატიკური ცნებები როგორიცაა ჰომომორფიზმი და დავამტკიცოთ მეტათეორემები. ჩვენ შეგვიძლია განვსაზღვროთ ენა, რომელიც იგივურია U^0-თან, როგორც ინდექსირებული ოჯახი $< A^{\ 0}, F_i^0, X\delta^{\ 0}, S^0,$ FOR> $i \in \{ 0 , 1 , 2 \} , \delta \in \Delta^{\ 0}$. იმისათვის რომ ეს ენა იყოს ცალსახა იგი უნდა აკმაყოფილებდეს ორ პირობას: სტრუქტურული ოპერაციებით მიღებული არცერთი გამოსახულება არ უნდა იყოს U^0-ის ძირითადი გამოსახულება. (ე. ი. $X\delta^{\ 0}$-ის ელემენტი) და ყოველი სწორი რთული გამოსახულება უნდა მიიღებოდეს მხოლოდ ერთი სტრუქტურული ოპერაციით და არგუმენტის მხოლოდ ერთი მნიშვნელობებით. მაგალითად. `( $P(a)$ ) $\wedge$ ( $Q(a)$) შეიძლება ჰქონდეს მხოლოდ ფორმა $F_2^{0(}$ `$\wedge$ ` , `P (a)`, `Q(a)`) U^0-ში და არ უნდა არსებობდეს სხვა ოპერაცია და არგუმენტები, რომლებიც მოგვცემს ამ ფორმულას. ეს იძლევა იმის გარანტიას, რომ ყოველ ფრაზას ექნება მხოლოდ ერთი ანალიზის ხე.

პრაგმატიკა (მონტეგიუ)

ენის შესწავლა (სემიოტიკა) მორისის [Foundations of the theory of Signs Chicaxa 1938] მიერ დაყოფილია სამ ნაწილად: სინტაქსი, სემანტიკა და პრაგმატიკა. სინტაქსი ეხება ლინგვისტურ გამოსახულებებს შორის დამოკიდებულებებს; სემანტიკა კი ეხება დამოკიდებულებებს გამოსახულებებსა და იმ ობიექტებს შორის, რომლებსაც ეს დამოკიდებულებები მიუთითებენ. პრაგმატიკა ეხება დამოკიდებულებებს გამოსახულებებსა, იმ ობიექტებსა, რომლებსაც ისინი მიუთითებენ, და გამომყენებლებს ან გამოსახულებათა გამოყენების კონტექსტებს შორის.

ენები და ინტერპრეტაციები

რაიმე პრაგმატული ენა განისაზღვრება როგორც ენა, რომელსაც აქვს შემდეგი კატეგორიის სიმბოლოები (ან ატომური გამოსახულებები):

(1) ლოგიკური კონსტანტები $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, Λ,V,=(შესამისად: `არაა შემთხვევა რომ`, `და`, `ან`, `თუ...მაშინ`, `მაშინ და მხოლოდ მაშინ`, `ყველასათვის`, `ზოგიერთისთვის`, `იგივურია`);

(2) მრგვალი ფრჩხილები;

(3) ინდივიდუალური ცვლადები: $V_{0,}$ $V_{1,}$...,$V_{m,...}$;

(4) n ადგილიანი პრედიკატები ყოველი ნატურალური რიცხვისათვის n, ერთადგილიანი E (არსებობს) პრედიკატის ჩათვლით;

(5) (n) ადგილიანი ოპერაციული სიმბოლოები ყველა ნატურალური რიცხვისათვის n;

(6) (n) ადგილიანი ოპერატორები ყოველი დადებითი მთელი n ისათვის.

ყოველი პრაგმატული ენისათვის მოითხოვება, რომ იგი შეიცავდეს (1), (2) და (3) კატეგორიის სიმბოლოებს. ამგვარად, პრაგმატული ენა არის საზოგადოდ პრედიკატების, ოპერაციების და ოპერატორების სიმრავლე.

რაიმე n ადგილიანი ოპერატორი არის რაიმე სიმბოლო, რომლის შემდეგ თუ ჩვენ მოვათავსებთ n წინადადებისაგან შედგენილ სტრიქონს, მოგვცემს ახალ წინადადებას. თერმები (აღმნიშვნელი გამოსახულებები) და ფორმულები განისაზღვრებიან ჩვეულებრივად რეკურსიული წესების გამოყენებით. L-ის თერმების სიმრავლე არის უმცირესი ისეთი U სიმრავლე რომ

(1) ყველა ცვლადები არიან U-ში და

(2) U შეიცავს $A\zeta_1...\zeta_n$-ს სადაც A- არის n ადგილიანი ოპერაციის სიმბოლო L-ში და $\zeta_1...\zeta_n$ ეკუთვნის U-ს. L-ის ფორმულების სიმრავლე არის უმცირესი ისეთი Δ სიმრავლე, რომ (1) Δ შეიცავს $P\zeta_1$ ...ζ_n –ს, სადაც P არის L-ის n ადგილიანი პრედიკატი და $\zeta_1...\zeta_n$ არიან L-ის თერმები და (2) Δ შეიცავს $\neg$, ($\varphi \wedge \Psi$) , ($\varphi \vee \Psi$) , ($\varphi \rightarrow \Psi$) და ($\varphi \equiv \Psi$) როცა φ და Ψ ეკუთვნის Δ -ს. (3) Δ შეიცავს $\Lambda_{u\varphi}$ და $V_{u\varphi}$, სადაც u არის ცვლადი და φ არის Δ -დან და (4) Δ შეიცავს $N\varphi_1...\varphi_n$-ს, როცა N არის L-ის n ადგილიანი ოპერატორი და $\varphi_{1...\varphi n}$ ეკუთვნის Δ -ს.

L-ის ინტერპრეტაციისას, ჩვენ უნდა გავითვალისწინოთ გამოყენების შესაძლო კონტექსტები. ამ კონტექსტებიდან უნდა ავარჩიოთ რელევანტური ასპექტების კომპლექსები და მათ ჩვენ ვუწოდებთ ინდექსებს ან კიდევ მითითების წერტილებს. მაგალითად, თუ L-ის ინდექსური თვისებებია მხოლოდ დროის ოპერატორების შეხვედრა, მაშინ მითითების წერტილები იქნებიან მომენტები, რომლებიც

შესა'ლებელია შეგვხვდეს გამოთქმებში როგორც დროის განსხვავებული მომენტები. თუ L შეიცავს აგრეთვე პირთა ნაცვალსახელებს, მაშინ მითითების წერტილები იქნებიან დალაგებული წყვილები, რომელთა ერთი კომპონენტი იქნება დროის მომენტი და მეორე კომპონენტი - პირი.

სხვა სახის ინფორმაცია, რომელიც დაგვჭირდება L-ის ინტერპრეტაციისათვის არის L -ის თვითეული P პრედიკატის ინტენსიონალი (მნიშვნელობა). ამისათვის, უნდა განისაზღვროს მითითების ყოველ წერტილში i P-ს ექსტენსიონალი (ანუ დენოტაცია). მაგალითად, თუ მითითების წერტილები არიან დროის მომენტები და P არის ერთადგილიანი პრედიკატი `არის თეთრი`, მაშინ დროის ყოველი მომენტისათვის უნდა დავადგინოთ სიმრავლე იმ ობიექტებისა, რომლებიც შეიძლება იყვნენ თეთრი.

მესამე სახის ინფორმაცია არის L-ის ყოველი ოპერაციის ინტენსიონალი, რომელიც ისევე შეი'ლება განისაზღვროს, როგორც პრედიკატის შემთხვევაში. მაგალითად, განვიხილოთ ერთადგილიანი ოპერაცია ``ვისიმე ქმარი``. დროის ყოველი მომენტისათვის არის ფუნქცია, რომელიც განსაზღვრავს ყოველი შეუღლებული ქალისათვის - მის ქმარს. მეოთხე სახის ინფორმაციაა L-ის ოპერატორების ინტერპრეტაცია. ამისათვის, L-ის ყოველ n ადგილიან ოპერატორთან მითითების i წერტილში ვაკავშირებთ ამ ოპერატორის ექსტენციონალს i-ს გათვალისწინებით, როგორც ამ ოპერატორის რაიმე n ადგილიან დამოკიდებულებას მითითების წერტილების სიმრავლეებს შორის. L-ის ინტერპრეტაციისათვის ჩვენ უნდა დავაზუსტოთ განსახილავი შესა'ლო ობიექტების სიმრავლე. ამისათვის შემოვიღოთ განმარტება.

განმარტება. I შესაძლო ინტერპრეტაცია L პრაგმატული ენისათვის არის დალაგებული სამეული < I , U, F> ისეთი ,რომ

(1) I და U არიან სიმრავლეები;

(2) F არის L-ზე განსაზღვრული ფუნქცია;

(3) ყოველი A სიმბოლოსათვის L-დან F_A არის I-ზე განსაზღვრული ფუნქცია;

(4) ყოველთვის როცა P არის L-ის n ადგილიანი პრედიკატი და $i \in I$, $E\, P_{(i)}$ არის n ადგილიანი დამოკიდებულება U-ზე;

(5) ყოველთვის, როცა A არის L-ის n ადგილიანი ოპერაცია და $i \in I$, $F_A(i)$ არის n+1 ადგილიანი დამოკიდებულება U-ზე ისეთი, რომ როცა $X_1, \ldots X_n$ ეკუთვნიან U-ს , არსებობს ზუსტად ერთი ობიექტი $Y \in U$ ისეთი, რომ $< X_1 \ldots, X_n, Y \in F_A(i)$, და

(6) ყოველთვის როცა N არის L-ის n-ადგილიანი ოპერატორი და $i \in I$, $F_n(i)$ არის n ადგილიანი დამოკიდებულება I-ის ყველა ქვესიმრავლეების სიმრავლეზე.

მნიშვნელობა და მითითება

იმისათვის, რომ დავახასიათოთ ჭეშმარიტება, ლოგიკური სისწორე (ვარგისიანობა, დასაბუთება) და ლოგიკური შედეგი შემოვიტანოთ ინტენსიონალისა და ექსტენსიონალის ცნებები. თერმის ექსტენსიონალი მოცემულ მითითების წერტილში არის ისეთი H ფუნქცია, რომელიც ნიშნავს მისი ცვლადების მნიშვნელობათა ყოველ შესაძლო სისტემისათვის ამ თერმის მნიშვნელობას მითითების მოცემულ წერტილში. თერმები შეიძლება შეიცავდნენ ცვლადების სხვადასხვა რაოდენობას, რომელიც არაა ზემოდან შემოსაზღვრული, ამიტომ მოსახერხებელია მივიღოთ, რომ H გამოიყენება შესაძლებელ ობიექტთა უსასრულო მიმდევრობებზე, ე. ი. H- აქვს არგუმენტების უსასრულო რაოდენობა. მიმდევრობის წევრის ნომერი ამყარებს თანადგომას მიმდევრობის წევრსა და თერმის შესატყვის ცვლადს შორის. ე.ი. მიმდევრობის k-ური წევრი არის თერმის k-ური ცვლადის მნიშვნელობა.

განსაზღვრება II. დავუშვათ C არის L პრაგმატული ენის რაიმე დასაშვები ინტერპრეტაცია

$C=< I , U, F>$ და $i \in I$. მაშინ $Ext_{i, C}(\zeta)$ ან ζ-ს ექსტენსიონალი i ში (C-თვის), სადაც Σ არის L-ის რაიმე თერმები, განისაზღვრება შემდეგი რეკურსიული განმარტებით:

(1) $Ext_{i,C}(V_n)$ არის ისეთი ფუნქცია H განსაზღვრული C-ს შესაძლო ობიექტების ყველა უსასრულო მიმდევრობათა სიმრავლეზე, რომ თუ X არის რაიმე ისეთი მიმდევრობა, მაშინ $H(X)=X_n$.

(2) თუ A არის L-ის n-ადგილიანი ოპერაცია და $\zeta_1,...,\zeta_n$ არიან L-ის თერმები, მაშინ

$Ext_{i,c}(A\zeta_1,... ,\zeta_n)$ არის ისეთი H ფუნქცია, რომლის განსაზღვრის არეა C-ს შესაძლო ობიექტების ყველა უსასრულო მიმდევრობათა სიმრავლე და X არის რაიმე ასეთი მიმდევრობა, მაშინ H (X) არის უნიკალური ობიექტი y, რომლისთვისაც

$<Ext_{i,c}(\zeta_1)X,...Ext_{i,c}(\zeta_n) X,Y >$ არის $F_A(i)$ ის წევრი.

განსაზღვრება III. დავუშვათ C არის L პრაგმატული ენის რაიმე დასაშვები ინტერპრეტაცია (I,U,F) და E არის L-ის თერმი. მაშინ Int $_C$(ζ) არის H ფუნქცია განსაზღვრის არით I ისე, რომ ყოველი i∈ I -სთვის

H(i)= Ext $_{i,c}$(ζ).

რადგანაც ფორმულის ექსტენსიონალი საზოგადოდ დამოკიდებულია ფორმულის ნაწილების ინტენსიონალზე, ამიტომ ფორმულის ექსტენსიონალის განსასაზღვრად პირველად უნდა განისაზღვროს ფორმულის ინტენსიონალი და შემდეგ მარტივი რეკურსიით შეი'ლება ფორმულის ექსტენსიონალის განსაზღვრა.

განსაზღვრება IV. დავუშვათ C არის L პრაგმატული ენის დასაშვები ინტერპრეტაცია < I , U, F > , მაშინ

Int $_{C,}$ სადაც φ არის L რაიმე ფორმულა, განისაზღვრება შემდეგნაირად:

(1) თუ ζ და η არიან L-ის თერმები, მაშინ Int$_c$(ζ=η)არის ისეთი H ფუნქცია განსაზღვრის არით I, რომ ყოველი i∈ I -სთვის H(i) არის U-ს წევრების X უსასრულო მიმდევრობათა სიმრავლე, რომლისთვისაც Ext $_{i,c}$(ζ)X იგივეა რაც Ext $_{i,c}$(η) (X).

(2) თუ P არის L-ის n ადგილიანი პრედიკატი და ζ$_1$,...,ζ$_n$ არიან L-ის თერმები, მაშინ Inti,cP(ζ$_{1,\ .\ .}$ ζ$_n$) არის ისეთი ფუნქცია H არით I, რომ ყოველი i∈ I -სთვის, H (i) არის U-ს წევრების X –უსასრულო მიმდევრობათა სიმრავლე, რომლისთვისაც < Ext $_{i,c}$ ζ$_1$(ζ$_n$)X,...,Ext $_{i,c}$(ζ$_n$)X> არის F$_p$(i)-ს წევრი.

(3). თუ φ არის L-ის ფორმულა, მაშინ Int$_c$(tφ) არის ისეთი ფუნქცია H არით I, რომ ყოველი i∈ I-სთვის, H(i) არის U-ს წევრების X უსასრულო მიმდევრობათა სიმრავლე, რომლისთვისაც X არის Int$_c$ (φ) (i) ში და ანალოგიურად განიმარტება სხვა დამაკავშირებლებისთვისაც.

(4) თუ φ არის L-ის ფორმულა, მაშინ Int$_c$(∨ V$_{nφ}$) არის ისეთი ფუნქცია H არით I, რომ ყოველი i∈I -სთვის H (i) არის U-წევრების X უსასრულო მიმდევრობების სიმრავლე, რომლისთვისაც არსებობს U-ში ისეთი Y, რომ უსასრულო მიმდევრობა < X$_{1,...,}$X$_n$, Y,X $_{n+1,...>}$ არის Int$_c$(φ) (i)-ში. ანალოგიურად განიმარტება ∧ V$_{nφ}$ -სთვის.

(5) თუ N არის n ადგილიანი ოპერატორი L-ში და φ 1, . . , φ$_n$ არიან L-ის ფორმულები, მაშინ Int$_c$(Nφ $_{1,\ .\ .\ ,\ φ\ n)}$ არის ისეთი ფუნქცია H არით I, რომ ყოველი L∈ I -თვის, H (i) არის U-ს წევრების X უსასრულო მიმდევრობათა სიმრავლე ისეთი , რომ < J$_1$,...,J$_n$ > არის F$_N$(i)-ში , სადაც ყოველი k<n-სთვის, J$_k$ -არის J წევრების სიმრავლე, რომლისთვისაც X არის Int$_c$ (φ $_k$) (j)-ს წევრი.

განსაზღვრება V. დავუშვათ, რომ C არის L-ის დასაშვები ინტერპრეტაცია, i არის C-ს მითითების წერტილი და φ არის L-ის ფორმულა, მაშინ Ext $_{i,c}$ (φ) არის Int$_c$ (φ) (i).

განსაზღვრება VI. თუ C არის L-ის დასაშვები ინტერპრეტაცია, i არის C-ს მითითების წერტილი და φ არის L-ის წინადადება, მაშინ φ არის ჭეშმარიტი i-ში მხოლოდ და მხოლოდ მაშინ, როცა $Ext_{i,c}$ (φ) არის C-ს შესა'ლო ობიექტების ყველა შესა'ლო მიმდევრობათა სიმრავლე.

განსაზღვრება VII. ჩვენ ვიტყვით, რომ φ არის რაიმე U სიმრავლის ლოგიკური შედეგი მაშინ და მხოლოდ მაშინ როცა არსებობს L პრაგმატული ენა ისეთი რომ φ და U-ს ყველა წევრები არიან L-ის წინადადებები და L-ის ყოველი დასაშვები ინტერპრეტაციისათვის C და C-ს ყოველი მითითების წერტილისათვის i, თუ U-ს ყველა წევრები არიან ჭეშმარიტი i-ში, მაშინ φ არის ჭეშმარიტი i-ში; და φ არის ლოგიკურად სწორი მაშინ და მხოლოდ მაშინ თუ φ არის ცარიელი სიმრავლის ლოგიკური შედეგი (ე. ი. მხოლოდ და მხოლოდ მაშინ თუ არსებობს L პრაგმატული ენა, რომელშიც φ არის წინადადება და ისეთი, რომ φ არის ჭეშმარიტი i-ში ყოველთვის როცა C არის L-ის დასაშვები ინტერპრეტაცია და i არის C-ს მითითების წერტილი).

სპეციალიზაციები

განვიხილოთ სპეციალური დისციპლინები, რომლებიც შედიან პრაგმატიკაში, როგორიცაა დროის ლოგიკა, მოდალური ლოგიკა და სხვები. ისინი წარმოადგენენ ინტერპრეტაციათა სპეციალურ კლასებს.

ჩვეულებრივი დროს ლოგიკა

განვიხილოთ L პრაგმატული ენა, რომელსაც აქვს მხოლოდ ერთადგილიანი ოპერატორები P და F ვთქვათ $K_1(L)$ იყოს < I ,U,C> შესაძლო ინტერპრეტაციების კლასი L-ისათვის ისეთი , რომ

(1a) I არის ნამდვილი რიცხვების სიმრავლე;

(1b) ყოველი i∈ I -სთვის, G_P (i) არის ერთეული წყვილების < J > სიმრავლე ისეთი რომ J⊆ I და არსებობს J ∈ J ისეთი, რომ J< i; და

(1c) ყოველი L ∈ I-სთვის G_F(i) არის ერთადგილიანი < ო > მიმდევრობების სიმრავლე Δ ისეთი, რომ J⊆ I და არსებობს J⊆ J ისეთი რომ i<J. აქ ჩვენ ვუყურებთ ნამდვილ რიცხვებს როგორც დროის მომენტებს. თუ (1a)- (1c) ადგილი აქვს, φ არის L-ის წინადადება და i∈ I ადვილია ვაჩვენოთ, რომ Fφ არის ჭეშმარიტი i-ში < I , U,G>

- მიხედვით მაშინ და მხოლოდ მაშინ როცა არსებობს J i-ში ისეთი, რომ J< i და φ არის ჭეშმარიტი J-ში < I , U,G > მიხედვით და Fφ არის ჭეშმარიტი i-ში < I,U,G> -ს მიხედვით მაშინ და მხოლოდ მაშინ როცა არსებობს J∈ I ისეთი რომ J > i და φ არის ჭეშმარიტი j-ში.

< I,U,G> -ს მიხედვით. ამგვარად, P და F ოპერატორები გამოსახავენ შესაბამისად წარსულ და მომავალ დროს და I სიმრავლე არის უწყვეტი სიმრავლე. თუ ჩვენ მოვითხოვთ რომ I სიმრავლე იყოს დალაგებული მაშინ მივიღებთ განზოგადოებულ დროის ლოგიკას. ბუნებრივ ენებში პირისა და ჩვენებითი ნაცვალსახელების შესასწავლად ჩვენ შეგვიძლია მოვითხოვოთ რომ L არ შეიცავდეს არცერთ ოპერატორს და ჰქონდეს განსაკუთრებული ნულადგილიანი ოპერაციული სიმბოლო **C,** და შესა'ლო ინტერპრეტაციებზე თუ დავადებთ გარკვეულ მოთხოვნებს ასეთი ლოგიკური ენა გამოიყენება პირისა და ჩვენებითი ნაცვალსახელების შესასწავლად. ასევე გარკვეული L ენის განხილვით და შესაძლო ინტერპრეტაციებზე სხვა შეზღუდვების დადებით მიიღებიან სტანდარტული და განზოგადოებული მოდალური ლოგიკები, სპეციალური დეონტიური ლოგიკა და სხვები.

ბუნებრივი ენის კომპიუტერული დამუშავება და პროლოგი

ბუნებრივი ენის კომპიუტერული დამუშავება პროლოგის საშუალებით შედგება შემდეგი ნაწილებისაგან. პირველ ნაწილში ხდება ბუნებრივი ენის გრამატიკების წარმოდგენა პროლოგ-პროგრამის სახით. ეს წარმოდგენა ბუნებრივად ხდება, რადგან გრამატიკის წესები შეიძლება პირდაპირ წარმოდგენილ იქნეს პროლოგის წესებისა და ფაქტების სახით და შემდეგ ანალიზი და სინთეზი შეიძლება ეფექტურად შესრულდეს პროლოგის გამოთვლის მექანიზმით. მეორე ანალიზის შედეგი შეიძლება წარმოდგენილ იქნეს სიური სტრუქტურის საშუალებით. შემდეგი ნაწილი არის ბუნებრივი ენის სემანტიკური წარმოდგენა ლოგიკაში. ეს წარმოდგენაც არის ბუნებრივი, რადგანაც პროლოგს უშუალო კავშირი აქვს ლოგიკასთან, ლოგიკური აქსიომები შეიძლება უშუალოდ წარმოდგენილ იქნეს პროლოგის წესების საშუალებით და პროლოგის გამოყვანის მექანიზმი უშუალოდ აღებულია ლოგიკიდან და ამიტომ კონკრეტული ლოგიკური გამოყვანის მექანიზმი (თუ იგი არ ემთხვევა პროლოგის გამოყვანის მექანიზმს) შეიძლება ტრანსლირებულ იქნეს ადვილად პროლოგის გამოყვანის მექანიზმში ისევ სპეციალური პროლოგ-პროგრამის საშუალებით. ეს იდეა იყო სწორედ ჩადებული პროლოგში მათი შემქმნელების მიერ (კოლმერაუერი [ ]). ჩვენი ძირითადი მიზანია განვიხილოთ

ბუნებრივი ენის ტექსტის სემანტიკის წარმოდგენის ლოგიკური ფორმის მქონე ენები. განვიხილავთ ორ ენას LFL-ს (ლოგიკურ ფორმიანი ენა) და DCG (განსაზღვრულ წინადადებიანი გრამატიკები). ეს ენები აღწერილი იქნება ქართული ენის წინადადებების ამ ენაზე წარმოდგენის მოცემით. ამგვარად, ბუნებრივი ენის კომპიუტერული დამუშავება ამ ენების გამოყენებით ხდება შემდეგნაირად: პირველად ინერება პროლოგ-პროგრამა ბუნებრივი ენის წინადადებებისათვის სინტაქსური წარმოდგენის მისაღებად, ე. ი. ამ პროგრამისათვის შესასვლელია ბუნებრივი ენის წინადადება და გამოსასვლელია ამ წინადადების სინტაქსური წარმოდგენა. შემდეგ ეს წარმოდგენა გადაიყვანება კვლავ ახალი პროლოგ-პროგრამით წინადადების სემანტიკური წარმოდგენის ენაზე. ე. ი. ახალი პროგრამის შესასვლელია სინტაქსური წარმოდგენა, ხოლო გამოსასვლელია ამ წინადადების სემანტიკური წარმოდგენა გამოსახული ლოგიკური ენის საშუალებით. შემდეგ ახალი პროლოგ- პროგრამით ხდება მიღებული წარმოდგენის გადაყვანა პროლოგ პროგრამაში და მისი შესრულება იძლევა საბოლოო შედეგს, რომელიც გათვალისწინებული იყო თავდაპირველი წინადადების კომპიუტერული დამუშავებით.

LFL ლოგიკური ენა

LFL-ს ჩვენ გამოვიყენებთ როგორც მნიშვნელობის წარმოდგენის ენას ბუნებრივი ენისათვის. წინადადებათა მნიშვნელობები არიან ლოგიკური ფორმები და ისინი არიან გამოსახულებები LFL-ში. მთავარი პრედიკატი LFL-ში არიან სიტყვის აზრები ბუნებრივ ენაში. სიტყვის აზრები შეესატყვისებიან სიტყვათა განსხვავებულ მნიშვნელობებს, რომლებიც მოცემულია სტანდარტულ ლექსიკონში. მაგალითის სახით განვიხილოთ ლოგიკური ფორმა წინადადებისათვის

პეტრეს ჰგონია, რომ ყოველ მამაკაცს უყვარს ხათუნა. (1) ეს წინადადება შეიძლება ჩაიწეროს ასე:

ჰგონია1 (პეტრე, ყოველი (მამაკაცი1 (X), უყვარს3 (X, ხათუნა))). (2)

სადაც ჰგონია1 არის ერთ-ერთი აზრი ზმნისა.

`` გონება``, მამაკაცი1 არის არსებითი სახელის ``მამაკაცი `` ერთ-ერთი აზრი და ა.შ.. გარდა ლექსიკური პრედიკატებისა, LFL-ში არის არალექსიკური პრედიკატებიც როგორიცაა ზმნის დროს მითითება ან კიდევ არსებითი სახელის რიცხვის მითითება და სხვა. მაგალითად,

პეტრემ შეიყვარა ხათუნა. (3)

შეიყვარა არის შეყვარება ზმნის წარსული დრო და იგი გამოისახება სპეციალური პრედიკატით.

წარსული- დრო (შეყვარება1 (პეტრე, ხათუნა)). (4)

შევნიშნოთ რომ მე-(2) წარმოდგენაში ყოველი არაა პრედიკატი, იგი არის ქვანტორი ``ყოველი ``, რომლის განსაზღვრის არეა ფრჩხილებში მოთავსებული ფორმა და ამ ქვანტორით დაბმული ცვლადია X. განვიხილოთ სხვა მაგალითი:

უყვარს პეტრეს ხათუნა? (5)

იგი წარმოიდგინება ასე:

პასუხი (true, უყვარს (პეტრე, ხათუნა)).

ამ პრედიკატის შესრულების შედეგია true თუ ბაზაში არსებობს ფორმა უყვარს (პეტრე, ხათუნა), წინააღმდეგ შემთხვევაში-false, სადაც true და false ლოგიკური კონსტანტებია. LFL-ის ყოველ პრედიკატს აქვს არგუმენტების ფიქსირებული რაოდენობა. არგუმენტებად შეიძლება იყვნენ ცვლადები, კონსტანტები ან რაიმე ლოგიკური ფორმები. აქედან გამომდინარე ლოგიკური თორმა ანუ LFL-ის გამოსახულება შეი'ლება განისაზღვროს ასე:

1. თუ P არის LFL-ის რაიმე პრედიკატი (სიტყვის რაიმე აზრი ან რაიმე არალექსიკური პრედიკატი), რომელსაც აქვს n არგუმენტი X1,...,Xn**,** სადაც თვითეული მათგანი არის რაიმე ცვლადი ან რაიმე კონსტანტა ან რაიმე ლოგიკური ფორმა (როგორც ეს მოითხოვება p-ს ამ არგუმენტისათვის), მაშინ p(X1,...,Xn) არის ლოგიკური ფორმა.

2. თუ P და Q ლოგიკური ფორმებია, მაშინ P&Q არის ლოგიკური ფორმა.

3. თუ P არის ლოგიკური ფორმა და E არის რაიმე ცვლადი, მაშინ P:E (იკითხება P ინდექსით E) არის ლოგიკური ფორმა.

`` : `` ოპერატორს ეწოდება ინდექსის ოპერატორი. როდესაც კონტექსტი იგნორირებულია ლოგიკური ფორმა P:E შეიძლება წარმოიდგინოს წინადადებით

P : P ← P ან რაც იგივეა P. სანამ სიტყვათა კლასების აღწერაზე გადავიდოდეთ განვიხილოთ სინტაქსის ზოგიერთი საკითხი, რადგან ლოგიკური ფორმა მჭიდროდაა დაკავშირებული სინტაქსთან.

ზმნები

წინადადების ძირითადი წევრია ზმნა. რადგან იგულისხმება რომ წინადადების ყოველ შემადგენელ ნაწილს აქვს მთავარი სიტყვა-წინადადებისათვის ასეთ მთავარ სიტყვას წარმოადგენს ზმნა. რაიმე წინადადებაში მთავარი ზმნა გარშემორტყმულია მოდიფიკატორებით როგორიცაა დამატებები და დამხმარე მოდიფიკატორები.

მაგალითად, სუბიექტი და ობიექტი არიან დამატებები, ხოლო ზმნისზედა არის დამხმარე მოდიფიკატორი. ლოგიკურ ფორმაში მთავარი ზმნა წარმოადგენს პრედიკატს, რომლის არგუმენტებია მისი დამატებები. ამგვარად სინტაქსური ცნება-დამატება მჭიდროდაა დაკავშირებული სემანტიკურ ცნებასთან-არგუმენტი. მაგალითად:

პეტრე მიდის. მისვლა1 (პეტრე).

პეტრე მიაცილებს თინას. მიაცილებს1(პეტრე, თინა).

პეტრე ჩუქნის ყვავილს თინას. ჩუქება1(პეტრე, ყვავილი, თინა).

შევნიშნოთ, რომ სიტყვათა ბრუნვის ნიშნები და ზმნის ფორმები არაა ცხადად მითითებული ლოგიკურ ფორმაში, რადგან სიტყვის ფუნქციას მკაცრად განსაზღვრავს არგუმენტის ადგილი ფორმაში, ხოლო ზმნის ფორმა ჩადებულია პრედიკატის მნიშვნელობაში. სიტყვის აზრი არის ინტენსიონალური თუ ამ სიტყვის ერთი არგუმენტი მაინც არის ლოგიკური ფორმა. მაგალითად

პეტრეს სჯერა, რომ თინა მოვა.

დაჯერება1 (პეტრე, მოსვლა2(თინა)).

ვნებითი გვარის კონსტრუქციები შეი'ლება იქნეს წარმოდგენილი შემდეგნაირად:

თინა არის მიცილებული პეტრეს მიერ.

პასივი (პეტრე, მიცილება (პეტრე, თინა)).

აქ პასივი (X,P) არის არალექსიკური პრედიკატი, სადაც P აღწერს თუ რა გადახდა პეტრეს. ზოგ შემთხვევაში შესაძლებელია პასივის იგნორირება და მაშინ გვექნება:

პასივი (X,P)← P

ზოგიერთ ზმნებს შეიძლება არ ჰქონდეს არგუმენტი. მაგ. თოვს. ასეთ შემთხვევაში გვაქვს უარგუმენტო პრედიკატი თოვს.

არსებითი სახელები

უმეტესობა არსებითი სახელებისა არიან ერთარგუმენტიანი პრედიკატები. მაგ. კაცი, რომელიც წარმოიდგინება კაცი (X), მაგრამ არიან არსებითი სახელები, რომლებიც აღნიშნავენ დამოკიდებულებას და ასეთ შემთხვევაში მათ შეესატყვისება ორარგუმენტიანი პრედიკატები. მაგ. მამა პრედიკატის წარმოდგენაა მამა(X,Y), რომელიც აღნიშნავს რომ X არის Y მამა. საკუთარი სახელები ხელს უწყობენ ცვლადის დაკავშირებას შესატყვის კონსტანტასთან. სხვა არსებით სახელთა ჯგუფებს აქვთ რაიმე დაკავშირებული ქვანტიფიკატორები, რომლებიც მოდიან წინადადების დარჩენილი ნაწილიდან. არსებითი სახელთა ჯგუფებისათვის

ქვანტიფიკატორები ძირითადად მოდის ზედსართავი სახელებისაგან და განმსაზღვრელებისაგან.

განმსაზღვრელი სახელები

არსებითი სახელებისათვის მოდიფიკატორების უმთავრესი ტიპები არიან განმსაზღვრელი სახელები, ზედსართავი სახელები და სხვა არსებითი სახელები. განმსაზღვრელ სახელთა რაოდენობა ფიქსირებულია. სემანტიკურად განმსაზღვრელები არიან ქვანტიფიკატორები. მაგალითებია: ზოგიერთი, ყოველი, მრავალი, ყველა, არცერთი, ცოტა, მისი, მათი, ეს, ის, ესენი, ისინი, რომელი, ვისი, რისი, ერთერთი და სხვა. მათ როგორც პრედიკატებს უმეტესად აქვთ არგუმენტი, რომლებიც არიან ლოგიკური ფორმები. პირველ არგუმენტს ეწოდება განმსაზღვრელის ბაზა, ხოლო მეორეს ფოკუსი. მათი ზოგადი სახეა:

განმსაზღვრელი (ბაზა,-ფოკუსი).

წყვილს (-ბაზა, -ფოკუსი) ეწოდება განმსაზღვრელის საზღვრები. მაგალითები:

ყოველ ადამიანს მოსწონს პეტრე.

ყოველი (ადამიანი (X), მოსწონს (X, პეტრე)).

ყოველ მამაკაცს უყვარს რომელიმე ქალი.

ყოველი (მამაკაცი (X), რომელიმე (ქალი (Y), უყვარს (X,Y))).

ყოველი მამაკაცი, რომელსაც უყვარს თინა, აძლევს მას საჩუქარს .

ყოველი (მამაკაცი (X)& საჩუქარი (Y)&, უყვარს (X,თინა), აძლევს (X,Y,თინა)).

სტანდარტულ უნივერსალურ ქვანტიფიკატორს საზოგადოდ აქვს სახე (ყველა (X,P) რაც ნიშნავს რომ ყოველი X-ისათვის ადგილი აქვს P-ს. LFL-ში ქვანტიფიკატორები განსხვავდებიან სტანდარტული განსაზღვრისაგან ორი რამით: (1) ქვანტიფიკატორის საზღვრები გახლეჩილია ორ ნაწილად - ბაზა და ფოკუსი და ჩვენ ვწერთ ყველა (P,Q) ნაცვლად ყველა (X,P→Q). ქვანტიფიკაცია შეიძლება შესრულდეს რამოდენიმე ცვლადზე ერთდროულად და ქვანტიფიკაცია ხდება ბაზის ყველა თავისუფალი ცვლადის მიხედვით. ეს კეთდება იმიტომ, რომ ბუნებრივი ენის ქვანტიფიკატორების გამოყენება განსხვავდება ლოგიკის სტანდარტული ქვანტიფიკატორისაგან. მაგალითად: ბევრ მამაკაცს მოსწონს მონრო.

ბევრი (მამაკაცი (X), მოსწონს (X, მონრო))

იმისათვის, რომ განვსაზღვროთ ``ბევრი`` უნდა ვიცოდეთ მამაკაცების რაოდენობა და მამაკაცების რაოდენობა, რომელთაც მოსწონს მონრო. ჩვენ უნდა განვსაზღვროთ card (P,N), სადაც N არის P-ების რაოდენობა

card (P,N)← set-of (P,D,S) & length (S,N).

set-of (X,Y,Z) არის ჩაშენებული პრედიკატი, სადაც Z არის იმ X -ების სია, რომლისთვისაც ადგილი აქვს Y-ს. length (S,N)-ში N გვაძლევს S სიაში ელემენტების რაოდენობას. ამის შემდეგ ბევრი (-ბაზა -ფოკუსი) შეიძლება განისაზღვროს ასე:

ბევრი (-ბაზა, -ფოკუსი)← card (ბაზა, A)& card (-ფოკუსი &-ბაზა, A1) & მეტი (A1,A).

აქ ქვანტორის საზღვრის ბაზად და ფოკუსად დაშლამ საშუალება მოგვცა ეფექტურად გაგვესაზღვრა IBM პროლოგზე ქვანტორი ბევრი. ბევრის ასეთ გაგებას ეწოდება ``ბევრი``-ს გაგება ფარდობითი აზრით, რაც ჩვენი მაგალითის შემთხვევაში ნიშნავს, რომ მონრო უფრო მეტ მამაკაცს მოსწონს, ვიდრე სხვა მამაკაცებს მოსწონს რომელიმე სხვა ქალი.

ნაცვალსახელები

ბევრი ნაცვალსახელი იქცევა ისევე როგორც განმსაზღვრელნი და ზოგიერთი მათგანი გრაფიკულადაც ემთხვევა განმსაზღვრელებს. ფორმალურად იმის დადგენა თუ კონკრეტული ნაცვალსახელი რომელ სახელს მიუთითებს არის ‘ნელი საკითხი და იგი საჭიროებს სპეციალურ შესწავლას და ამ მიმართულებით მრავალი გამოკვლევები არსებობენ, ხოლო ქართული ენისათვის ასეთი გამოკვლევები არაა ჩატარებული. ნაცვალსახელები ლოგიკურ ფორმაში შეიძლება მიუთითებდნენ ცვლადებს, კონსტანტებს ან ლოგიკურ ფორმებს. მაგალითად.

პეტრეს ჰყავს ძაღლი. მას უყვარს იგი.

ძაღლი (X) & ჰყავს (პეტრე, X) & უყვარს (პეტრე, X).

ყოველ ადამიანს უყვარს თავისი თავი.

ყოველი (ადამიანი (X), უყვარს (X,X))

ზმნისზედები

ზმნისზედები ჰქმნიან სიტყვათა ღია კლასს, რომლებიც აზუსტებენ ზმნებს, ზედსარვავ სახელებს და სხვა ზმნისზედებს. ისინი არიან ინტენსიონალურები. ზოგიერთებს აქვთ ერთი არგუმენტი და აზუსტებენ არგუმენტის ლოგიკური ფორმის დროს. მაგალითად,

გუშინ პეტრემ იყიდა ვაშლი

გუშინ (ex (ვაშლი (x), ყიდულობს (პეტრე, x)))

ფოკუსის ფუნქცია კარგად ჩანს შემდეგ მაგალითებში:

პეტრე ყოველთვის ყიდულობს წიგნებს პავლესთან.

ყოველთვის (წიგნი (x) & ყიდულობს (პეტრე, x)):E, თან (პავლე,E)).

პეტრე ყოველთვის ყიდულობს წიგნებს პავლესთან.

ყოველთვის (თან (პავლე, ყიდულობს (პეტრე, x)), წიგნი (x)).

ამ მაგალითებიდან ჩანს, რომ ერთიდაიგივე სიტყვებისაგან შედგენილი წინადადებები რომლებიც განსხვავდებიან მხოლოდ ფოკუსით აქვთ განსხვავებული ლოგიკური ფორმები. აღმატებითი ხარისხებიანი ზმნისზედებისათვის ლოგიკური ფორმები მიიღებიან ანალოგიურად იმისა, რაც იყო გაკეთებული ``ბევრი `` -სთვის.

ზედსართავი სახელები

ზედსართავები ჰქმნიან ღია კლასს და მოდიფიკაციას უკეთებენ არსებით სახელებს. შეი'ლება გამოვიდნენ დამატების როლში ზოგიერთი ზმნებისათვის როგორიცაა ``ყოფნა`` და ``ჩანს``. ზედსართავიდან მიიღებიან აგრეთვე ზმნისზედები. უმარტივესი ზედსართავები სემანტიკურად არიან ექსტენსიონალურები. მაგალითად,

პეტრე ყიდულობს წითელ ვაშლს.

ex (ვაშლი (x)& წითელი (x), ყიდულობს (პეტრე,x)).

ინტენსიონალურ ზედსართავ სახელებს აქვთ ერთი არგუმენტი, რომელიც მოდის ძირითადი არსებითი სახელიდან ან სხვა მოდიფიკატორიდან. მაგალითად,

პეტრე იცნობს სახელგანთქმულ ფეხბურთელს.

ex (სახელგანთქმული (ფეხბურთელი (x)), იცნობს (პეტრე, x))

პეტრე არის ერთადერთი შვილი

ერთადერთი (შვილი (x, პავლე), x=პეტრე)

თანდებულები

თანდებულები ჰქმნიან სიტყვათა ჩაკეტილ კლასს და შემოჰყავთ თანდებულიანი ფრაზები. ესენი არიან არსებით სახელიანი ფრაზები დაბოლოებულნი თანდებულებით. თანდებულთა აზრი არის ორადგილიანი პრედიკატი. პირველი

არგუმენტი არის ის არსებით სახელიანი ფრაზა, რომელსაც ახლავს თანდებული და მეორე არგუმენტი, ფრაზა რომელიც დაზუსტებულია ამ თანდებულიანი ფრაზით. მაგალითები:

პეტრე მიდის სახლში

ში-ადგილი (სახლი (x), მიდის (პეტრე))

პეტრე მიდის თბილისში

ში-ადგილი (პეტრე, მიდის (პეტრე))

კავშირები

კავშირები ჰქმნიან სიტყვათა ჩაკეტილ კლასს. კავშირები არიან ორი სახის: მაკოორდინებელი და დაქვემდებარებული კავშირები. კავშირები არგუმენტებად იღებენ ორ ლოგიკურ ფორმას. მაგალითები:

პეტრე აღებს კარებს და შედის ოთახში.

და (კარები (x) & აღებს (პეტრე,x), ში-ადგ (ოთახი (Y), შედის (პეტრე)))

ამ შემთხვევაში ``და`` არაა იგივე რაც კონიუნქცია, რადგან იგი გამოსახავს აგრეთვე მოქმედებათა თანმიმდევრობას. მაქვემდებარებელი კავშირების მაგალითებია:

როცა მაღლი ავია იგი ყეფს.

როცა (მაღლი (X) & ავი (X), ყეფს (X))

თუ A> B მაშინ B< A

თუ-მაშინ (მეტი (A , B) , ნაკლები (B , A))

არალექსიკური პრედიკატები LFL-ში

ზოგიერთი არალექსიკური პრედიკატები დაკავშირებულია სიტყვების ფლექსიურ ცვლილებებთან, როგორიცაა გრამატიკული რიცხვი და დრო. მაგალითად, past (p) ნიშნავს, რომ P ლოგიკური ფორმა განხილულია წარსულ დროში. არალექსიკურ პრედიკატებს მიეკუთვნება აგრეთვე პრედიკატი ``კი არა `` , რომელიც დასმულ კითხვაზე იძლევა პასუხს კი ან არა. მაგალითად,

ნახა პეტრემ ნანა გუშინ?

კი არა (გუშინ (ნახულობს (პეტრე, X), X=თინა)).

მრავალი ლექსიკური პრედიკატი წარმოიშვება ქართული ზმნების მორფოლოგიური კატეგორიების დასადგენად.

მაინდექსირებელი ოპერატორი

ბუნებრივი ენის წინადადებები აღწერენ სიტუაციებს, რომლებიც დაკავშირებულია დროსთან და ადგილთან. ე. ი. მათ მიერ აღწერილი მოქმედებები მიმდინარეობენ გარკვეულ დროში ან ადგილზე ან სხვა კონკრეტულ სიტუაციაში. მაგალითად,

პეტრემ ირბინა უსწრაფესად გუშინ.

უსწრაფესად (დარბის (პეტრე) :E გუშინ (E)).

აქ E აღნიშნავს დროის მომენტს გუშინ და ჩანაწერი ამბობს რომ პეტრემ გაირბინა უსწრაფესად E მომენტში და E-ს მნიშვნელობაა გუშინ.

პეტრემ დაინახა თინა. ეს მოხდა 11 საათამდე.

ხედავს (პეტრე, თინა) :E & მდე (11:00,E).

აქ წარსული დრო არაა განხილული სიმარტივისათვის, როცა ჩვენ გვაქვს ისეთი სამყარო, სადაც კონტექსტი არავითარ როლს არ თამაშობს, მაშინ ჩვენ შეგვი'ლია მივიღოთ:

P:P← P.

როცა მხოლოდ დროს აქვს მნიშვნელობა, მაშინ P : E შეიძლება ჩავწეროთ პროლოგზე როგორც ადგილი აქვს (P,T), სადაც T არის დროის მომენტი E. ასევე,

შეიძლება გამოვიმუშაოთ სპეციალური წესები ინდექსირებული პრედიკატების პროლოგში გადასაყვანად ყოველ განსაკუთრებულ შემთხვევებში.

გამოყენებული ლიტერატურა

[1]. A.Aho, J.Ullman. The Theory of Parsing, Translation and Compiling, vol. 1, Prentice-Hall, N. J., 1972 (Русский перевод В.Н. Агафонова, Под редакцией В.М. Курочкиа, Теория Синтаксического Анализа Перевода и Компилации, Том 1, Синтаксический Анализ, Москва, 1978).

[2]. R.Montague. The Proper Treatment of Quantification in Ordinary English, in K.J.J.HintikkaJ.M.E.Moravcsik and P.Suppes(eds), 'Approach to Natural Language', Synthese Library 49, Reidel, Dordrecht, 1973.

[3]. G.Gazdar, E.Klein, G.Pullum, I.Sag. Generalized Phrase Structure Grammar, Oxford University Press, Oxford, 1985

[4]. G.Erbach. A Bottom-up Algorithm for Parsing and Generation, CLAUS report, number 5, Saarland University, 1991

[5]. M.Kay. Algorithm Schemata and Data Structure in Syntactic Processing, Report SLS-12-80, Palo Alto, 1980

[6]. A.Joshi. Tree-Adjoining Grammars, in The encyclopedia Language and Linguistics, R.E.Asher(ed.), Pergamon Press, Oxford, UK, 1994

[7]. A.Aho. Nested Stack Automata, Journal of the ACM, Vol. 16, Issue 3, 1969

[8]. R.Sharp. CAT2: An Experimental EUROTRA Alternative, Machine Translation, Volume 6, Number 3, Springer Netherlands, 1991

[9]. R.Jonson, M.Rosner. A Rich Environment for Experimentation with Unification Grammars, in Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics

[10]. R.Kasper. A Unification Method for Disjunctive Feature Descriptions, in Proceedings of the 25 Annual Meeting of the Association for Computational Linguistics

[11]. R.Kaplan, J.Bresnan. A Formal System for Grammatical representation , in Formal Issues in Lexical functional Grammar, CSLI Lecture Notes, N47

[12]. G.Pollard, I.Sag. Head Driven Phrase Structure Grammar, CSLI and University Chicago Press, Stanford/Chicago, 1994

[13]. B.Partee. Montague Grammar and Transformational Grammar, Linguistic Inquiry 6, 1975.

[14]. R.Thomason(ed.), Formal Philosophy. Selected Papers of Richard Montague, Yale University Press, New Haven, 1974

[15]. J.J.Katz, J.A.Fodor.The structure of a semantic theory (1963)

[16]. T.M.V.Jansen. Foundations and Applications of Montague Grammar, Part 2: Application to Natural Language, CWI Tract 28, 1988

[17]. H. Kamp. A Theory of Truth and Semantic Representation, inJ. Groenendijk, Th. Janssen, and M. Stokhof, editors, Formal Methods in the Study of Language, Mathematisch Centrum,Amsterdam, 1981

[18]. C.Morris. Foundations of the Theory of Signs, in International Encyclopedia of Unified Science 1, 1938

[19]. A.Colmerauer. Prolog in 10 figures, Communication of the ACM, Volume 28, Issue 12, 1985

[20]. C.Beirle, J. Dorre, U. Pletat, C. Rollinger, P. Schmitt, R.Sfuder. The Knowledge Representation Language LILOG , CSL 88,1989

[21]. N. Chomsky. Semantic Structures, The Hague : Mouton,1957

[22].W.A.Woods. AN Experimental Parsing System for Transition Network Grammars, Massachusetts, 1972.

[23].S.M.Chou and K.S. FU.. Transition Network for pattern Recognition, Tech.Rept.TR-EE 75-39, Purdue university, 1975.

[24]. J.Antidze, D.Mishelashvili. Morphological and Syntactic Analysis of Natural Language Texts, Internet Academy, Georgian Electronic Scientific Jurnals: Computer Sciences and Telecommunications, 1(12), 2007, IISN 1512-1232, (in English). http://gesj.internet-academy.org.ge/gesj_articles/1345.pdf

[25]. Ginsburg, Seymour (1975). Algebraic and automata theoretic properties of formal languages. North-Holland. pp. 8–9. *ISBN 0-7204-2506-9*.

[26]. Harrison, Michael A. (1978). Introduction to Formal Language Theory. Reading, Mass.: Addison-Wesley Publishing Company. p. 13. *ISBN 0-201-02955-3*.

[27]. Sentential Forms, Context-Free Grammars, David Matuszek

[28]. Earley, Jay, "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM*, Vol. 13 No. 2, pp. 94-102, February 1970.

[29].Joshi, Aravind K., *et al.*, "Tree Adjunct Grammars," *Journal of Computer Systems Science*, Vol. 10 No. 1, pp. 136-163, 1975.

[30]. Knuth, Donald E., "Semantics of Context-Free Languages," *Mathematical Systems Theory*, Vol. 2 No. 2, pp. 127-145, 1968.