Tema 1 LFA 2015-2016
Gramatici independente de context

George Daniel MITRA

28 noiembrie 2016

Rezumat

Tema constă în implementarea unui program care analizează o gramatică independentă de context.

1 Specificații temă

1.1 Cerință

Să se implementeze un program care, primind o reprezentare a unei gramatici independente de context, să răspundă la următoarele întrebări despre ea:

- Este limbajul generat de gramatică vid?
- Se află șirul vid în limbajul generat de gramatică?
- Care sunt simbolurile neterminale inutile?

Soluțiile care nu folosesc FLEX vor fi depunctate după cum e menționat în secțiunea 4.

1.2 Conținutul arhivei

Arhiva trebuie să conțină:

- surse, a căror organizare nu vă e impusă
- un fișier Makefile care să aibă target de build și run (Puteți folosi o versiune modificată a celui din tema 0)
- un fișier README care să conțină numele și grupa și maxim 20 de linii de maxim 80 de caractere în care să menționați algoritmul aplicat pentru construcția greamaticii și să descrieți abordarea pentru lexer. Cu cât mai scurt, cu atât mai bine!

Nerespectarea oricărui aspect mai sus menționat va duce la nepunctarea temei.

Arhiva trebuie să fie zip. Nu rar, 7z, ace sau alt format ezoteric. Fișierul Makefile și fișierul README trebuie să fie în rădăcina arhivei, nu în vreun director.

1.3 Format README

Prima linie din README trebuie să conțină numele vostru, așa cum apare pe cs.curs.pub.ro. Prenumele trebuie să fie complet, numele să fie scris cu majuscule. Dacă aveți diacritice în nume, ele trebuie să apară și salvați fișierul ca UTF-8. Prima linie ar trebui să arate așa, cu mențiunea că folosiți numele vostru:

Nume: MITRA George Daniel

A doua linie trebuie să conțină seria și grupa. Seria și grupa trebuie să fie lipite, fără spații între ele, seria apărând înainte. Exemplu:

Grupa: CB334

Restanțierii de anul patru își vor scrie grupa curentă, nu grupa în care erau când au făcut LFA, punând un R în față. Exemplu:

Grupa: R342C4

Restul textului trebuie să încapă în maxim 20 de linii și fiecare linie să aibă cel mult 80 de caractere.

Scrieți doar ceea ce e esențial!

Fișierul trebuie să se numească README, nu readme, ReadMe, README.txt, readme.txt, read-me.doc, rEADME, README.md sau alte variante asemănătore sau nu. Nerespectarea acestor cerințe duce la nepunctarea temei.

1.4 Specificații program

1.4.1 Intrări

Programul va citi dintr-un fișier numit „grammar” gramatica independentă de context, în formatul specificat mai jos.

Programul va primi ca argument în linia de comandă întrebarea la care trebuie să răspundă:

- --is-void : Verifică dacă limbajul generat de gramatică este vid
- --has-e : Verifică dacă limbajul generat de gramatcă conține șirul vid
- --useless-nonterminals : Afișează care sunt simbolurile neterminale inutile.

Intrările nu se consideră corecte. Programul trebuie să raporteze orice problemă

1.4.2 Ieșiri

În cazul în care intrările sunt corecte, programul afișează răspunsul la ieșirea standard și nu afișează nimic la ieșirea de eroare.

Dacă argumentul în linia de comandă este --is-void sau --has-e, atunci programul afișează „Yes” sau „No”, în funcție de răspuns.

Dacă argumentul în linia de comandă este --useless-non-terminals, atunci programul afișează fiecare simbol neterminal inutil pe câte o linie. Ordinea nu contează. Un simbol neterminal este considerat inutil dacă nu există nicio secvență de derivări care pornind de la el să ajungă la un șir de terminali.

1.4.3 Erori

În cazul în care intrările nu sunt corecte, programul afișează eroarea la ieșirea de eroare și nu afișează nimic la iesirea standard.

În cazul în care argumentul în linie de comandă lipsește sau nu este unul din cele trei menționate în subsecțiunea 1.4.1, programul afișează mesajul „Argument error". Erorile de argument au cea mai mare prioritate

În cazul în care gramatica nu respectă sintaxa specificată în secțiunile 2.2 și 3.3, programul afișează mesajul „Syntax error". Dacă fișierul lex e bine făcut, cazurile de erori sintactice se rezolvă cu o singură regulă la final. Erorile sintactice au prioritate mai mare decât erorile semantice.

În cazul în care automatul are erori semantice, programul afișează mesajul „Semantic error". Erorile semantice au prioritate mai mare decât răspunsurile la întrebări.

2 Noțiuni introductive

2.1 Limbajul de descriere

Limbajul este descris printr-o gramatică BNF și folosește următoarea convenție de culori:

- albastru - neterminali
- verde - operatori ai limbajului BNF și paranteze ajutătoare
- rosu - terminali (elemente care fac parte efectiv din limbajul descris)

Pentru a simplifica sintaxa, Se folosesc operatorii $*$, $+$ și ? cu semnificația din expresiile regulate.

2.2 Simbol, Alfabet, Șir

2.2.1 Simbol

Un terminal poate fi literă, cifră sau caracter special.

Atenție, se folosesc simboluri diferite față de tema 0:

<terminal> ::= <lower-case letter> | <digit> | <other>

<lower-case letter> ::= (a | b | c | d | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z)

<digit> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)

<other> ::= (‘ | - | = | [|] | ; | ´ | \ | . | / | ˜ | ! | @ | # | $ | % | ˆ | & | ∗ | _ | + | : | ¨ | | | < | > | ?)

<other> reprezintă toate simbolurile printabile care nu sunt alfanumerice de pe o tastatură standard cu layout US International. Nu copiați din document simbolurile, pentru că s-ar putea să nu se potrivească.

2.2.2 Alfabet

Un alfabet este orice mulțime de terminali.

Atenție, alfabetul poate fi vid și apare ca {} în cazul ăsta.

<alphabet> ::= { (<terminal> (, <terminal>)*)? }

2.2.3 Șir

Un șir este o secvență finită de terminali. Șirul vid se notează cu e, în timp ce alte șiruri reprezintă o concatenare de unul sau mai multe simboluri. Din acest motiv, e nu se consideră simbol, deci nu va face niciodată parte din niciun alfabet sau șir.

<word> ::= e | (<terminal>)+

3 Gramatici independente de context

3.1 Descriere

O gramatică este, din punct de vedere formal, un tuplu $G = (V, \Sigma, R, S)$, unde V reprezintă mulțimea simbolurilor ce pot apărea în reguli, Σ reprezintă alfabetul, R reprezintă mulțimea regulilor și S reprezintă simbolul de start.

Σ este o mulțime finită de simboluri care pot apărea în cuvintele limbajului, simboluri numite și terminali. V este mulțimea tuturor simbolurilor care apar în reguli, dar, pe lângă terminali, conține și simboluri care nu apar în cuvintele limbajului, numite neterminali. Un astfel de simbol este $S \in V \setminus \Sigma$.

R este mulțimea finită a regulilor de producție. O regulă este un tuplu, $(A, \gamma), A \in V \setminus \Sigma, \gamma \in V^*$, cu semnificația că neterminalul A poate fi înlocuit cu șirul de terminali și neterminali γ. Modul în care este definit face ca $R \subset (V \setminus \Sigma) \times V^*$.

Înlocuirea părții din stânga a unei reguli cu partea din dreapta într-un șir se numește derivare. Spunem că v este derivat din $u (u \Rightarrow v)$ dacă $\exists A \in V \setminus \Sigma, \exists \alpha, \beta, \delta \in V^*$ astfel încât $(A, \delta) \in R \wedge u = \alpha A \beta \wedge v = \alpha \delta \beta$.

Închiderea reflexivă și tranzitivă a relației de derivare se notează cu $\Rightarrow^*$. $u \Rightarrow^* v$ înseamnă că v este derivat din u în zero sau mai mulți pași.

O gramatică generează un șir w dacă și numai dacă pornind de la simbolul de start se poate deriva w. Mulțimea tuturor șirurilor generate de o gramatică G se numește limbaj generat de gramatică. $w \in \mathcal{L}(G) \Leftrightarrow S \Rightarrow^* w$.

3.2 Detaliere cerințe

3.2.1 Stabilirea dacă limbajul generat de gramatică conține șirul vid

$e \in \mathcal{L}(G) \Leftrightarrow S \Rightarrow^* e$

Fie $r = (A, \gamma) \in R$ o regulă oarecare. Pentru ca pornind de la A să se poată deriva e folosind r, trebuie să se întâmple două lucruri:

1. $\gamma \in (V \setminus \Sigma)*$. Regula nu are voie să conțină terminali în partea dreaptă
2. $\forall B \in V \setminus \Sigma, \forall \alpha, \beta \in V^*, (\gamma = \alpha B \beta \rightarrow B \Rightarrow^* e)$. Pornind de la orice neterminal din regulă trebuie să se poată deriva șirul vid.

Pentru a implementa, se pot parcurge toate regulile marcând succesiv neterminali din care se poate deriva șirul vid. Dacă există cel puțin o regulă a cărei parte dreaptă este formată doar din neterminali marcați, atunci neterminalul din stânga regulii poate fi marcat. Marcarea se oprește în momentul în care după o parcurgere completă a mulțimii regulilor nu se mai poate marca niciun neterminal. Pentru timp mai scurt puteți aplica algoritmul după eliminarea simbolurilor inutile.

3.2.2 Eliminarea simbolurilor inutile din gramatică

În același fel se pot marca simbolurile utile. Un simbol este util dacă pornind de la el se poate deriva un șir. Un simbol este inutil dacă nu este marcat ca util. Orice regulă care conține un simbol inutil poate fi eliminată pentru că din acel simbol nu se va putea niciodată deriva un șir.

3.2.3 Stabilirea dacă limbajul generat de gramatică este vid

Limbajul generat de gramatică e vid dacă și numai dacă simbolul de start este inutil.

3.3 Specificații

În temă, o gramatică independentă de context este dată ca un tuplu, conform definiției:

<CFG> ::= (<(non)terminals> , <alphabet> , <production rules> , <start symbol>)

<(non)terminals> ::= { <(non)terminal> (, <(non)terminal>)* }

<(non)terminal> ::= (<nonterminal> | <terminal>)

<nonterminal> ::= <upper-case letter>

<production rules> ::= { (<production rule> (, <production rule>)*)? }

<production rule> ::= (<nonterminal> , <replacement>)

<replacement> ::= e | (<(non)terminal>)+

<start symbol> ::= <upper-case letter>

<upper-case letter> ::= (A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z)

Atenție, mulțimea regulilor de producție poate fi vidă și apare ca {} în fișierul de intrare.

3.4 Erori semantice

Pentru ca gramatica să fie corect definită există niște restricții:

- $\Sigma \subset V$. Toți terminalii se află în mulțimea de terminali și neterminali
- $V \setminus \Sigma$ e format doar din neterminali sau toți terminalii din V sunt și în Σ
- $S \in V \setminus \Sigma$. Simbolul de start se află în mulțimea terminalilor si neterminalilor
- $\forall (A, \gamma) \in R, A \in V \setminus \Sigma$. Partea din stânga a unei reguli se află în mulțimea terminalilor si neterminalilor
- $\forall (A, \gamma) \in R, \gamma \in V^*$. Toți terminalii sau neterminalii din partea dreaptă a oricărei reguli se află în mulțimea terminalilor si neterminalilor

Nerespectarea oricărei dintre restricțiile de mai sus necesită afișarea unei erori semantice.

Mulțimile nu conțin duplicate, deci nu e nevoie să se verifice asta.

4 Punctaj

4.1 Checker

Checker-ul oferă un punctaj între 0 și 200. Nu există bonus. Testele sunt publice.

4.2 Depunctări

Implementările care nu folosesc flex primesc maxim 30% din punctaj.

Pentru temele care nu respectă specificațiile la rularea automată, punctajul va fi 0.

Deadline: 23.12.2016, 23:59. Upload-ul va rămâne deschis pâna la ora 05:00.

5 Sugestii

Vă recomand să începeti cât mai repede. E mai ușor decât credeti. Timp estimat de lucru: 20 de ore.

Bibliografie

[1] flex homepage

[2] Lexical Analysis with Flex

[3] Using flex

[4] jflex homepage

[5] jflex user manual

[6] jflex user manual in japanese

[7] Laborator 1 SO: Makefile