Tema 2 LFA
Limbaje regulate - Reprezentare ca XML

George Daniel MITRA

6 ianuarie 2014

Rezumat

Tema constă în implementarea unui program care trece o formă de reprezentare finită a limbajelor regulate din specificația temei 1 în XML și invers.

1 Specificații temă

1.1 Cerință

Să se implementeze un program care, primind o formă de reprezentare finită a unui limbaj regulat, să îl convertească în XML și invers.

Programul trebuie să folosească FLEX pentru parsarea intrării, ca în tema 1. Soluțiile care nu folosesc FLEX vor fi depunctate după cum e menționat în secțiunea 6.

Update 07.01.2014;01:51; Pentru partea de automate finite XML puteți să faceți fără FLEX. Dacă alegeți asta, explicați în README soluția și avantaje/dezavantaje față de FLEX.

1.2 Limbaje de programare

Pentru această temă se pot folosi C/C++, Java sau Haskell. Dacă aveți o altă propunere pentru care există un generator de parsere de tipul flex, postați pe forum și dacă e suficient de similar vă vom aproba utilizarea.

1.3 Conținutul arhivei

Arhiva trebuie să conțină:

- surse, a căror organizare nu vă e impusă de noi
- un fișier Makefile care să aibă target de build.
- un fișier README, în care să descrieți abordarea pentru lexer/parser.

1.4 Specificații program

1.4.1 Parametri în linie de comandă

Programului i se vor da parametri în linie de comandă. Ordinea parametrilor nu contează. Parametrii descriu intrarea, fiindcă nu se cer conversii între forme de reprezentare.

Parametrii care descriu intrarea sunt următorii. Unul singur va apărea:

- --from-RE: Specifică faptul că se va citi o expresie regulată, în formatul specificat mai jos
- --from-DFA: Specifică faptul că se va citi un automat finit determinist
- --from-NFA: Specifică faptul că se va citi un automat finit nedeterminist

Opțional, pe lângă parametrul de intrare obligatoriu, poate apărea și:

- --from-XML: Specifică faptul că se va citi în formatul XML
- --to-XML: Specifică faptul că se va afișa în format XML

Pentru tema 1.5, poate apărea un singur parametru de ieșire:

- --to-RE: Specifică faptul că se va afișa o expresie regulată
- --to-DFA: Specifică faptul că se va afișa un automat finit determinist
- --to-NFA: Specifică faptul că se va afișa un automat finit nedeterminist
- --contains: Permite interogări ale apartenenței cuvintelor la limbajul descris. Parametrii de după „--contains" sunt cuvintele căutate. Pot fi zero sau mai multe cuvinte.

1.4.2 Intrări

Programul va citi de la intrarea standard expresia, automatul sau gramatica, în formatul specificat în secțiunea alocată fiecăruia.

Se consideră corect orice primește programul ca intrare.

Se ignoră orice fel de whitespace.

1.4.3 Ieșiri

Programul va afișa la ieșirea standard rezultatul, în formatul corespunzător. Rezultatul trebuie să fie urmat de o linie goală.

Pentru tema 1.5, în cazul unei interogări de tipul „--contains", pentru fiecare cuvânt se va afișa pe câte o linie „True", respectiv „False", dacă limbajul conține cuvântul sau nu.

2 Noțiuni introductive

2.1 Limbajul de descriere

Limbajul este descris printr-o gramatică BNF și folosește aceeași convenție de culori ca articolul Wikipedia despre BNF:

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

2.2 Simbol, Alfabet, Șir, Limbaj

2.2.1 Simbol

Un simbol este un caracter, semnal sau orice obiect ce poate fi interpretat cumva. Exemple de simboluri sunt în secțiunea 2.2.2

În temă, un simbol poate fi literă, cifră sau caracter special:

<symbol> ::= <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> ::= (! | # | $ | % | & | - | . | / | : | ; | < | > | = | @ | [|] | ^ | ‘ | ~)

Pentru XML, formatul de reprezentare pentru un simbol este:

<symbol-XML> ::= <symbol> <symbol> </symbol>

Exemplu pentru simbolul p: <symbol> p </ symbol>

2.2.2 Alfabet

Un alfabet este orice mulțime nevidă finită de simboluri.

În limbajul temei, un alfabet conține cel puțin un simbol, deci se declară astfel:

<alphabet> ::= { <symbol> (, <symbol>)* }

Pentru XML, formatul de reprezentare este:

<alphabet-XML> ::= <alphabet> (<symbol-XML>)+ </alphabet>

2.2.3 Șir

Un șir(cuvânt) este o secvență finită de simboluri dintr-un alfabet. Un șir poate să nu conțină niciun caracter. Acesta, șirul vid, se notează cu e(în alte surse se notează cu ε).

În limbajul temei, 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.

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

Pentru XML, formatul de reprezentare este:

<word-XML> ::= <word> (e | (<symbol-XML>)+) <word>

Exemplu pentru șirul ul:

<word> <symbol> u </symbol> <symbol> l </symbol> </word>

3 Expresii Regulate(ER)

3.1 Specificații

În temă, din cauza faptului că le folosim pentru a desemna elemente constitutive expresiilor regulate, caracterele $\{|, *, +, ?, O, e, (,)\}$ nu pot face parte din niciun alfabet. O expresie regulată se definește în felul următor:

<RE> ::= <alphabet> : <expression>

<expression> ::= O | e | <symbol> | (<expression> | <expression>) | (<expression> <expression>) | (<expression> *) | (<expression> +) | (<expression> ?) | (<expression>)

În temă, | reprezintă reuniunea, O reprezintă expresia limbajului vid, e reprezintă expresia limbajului care conține doar șirul vid.

Pentru XML, fiecare operație se va reprezenta cu câte o pereche de taguri:
<RE-XML> ::= <RE> <alphabet-XML> (<Reunion-XML> | <Concatenation-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>) </RE>

<Reunion-XML> ::= <reunion> (<Concatenation-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>) (<Concatenation-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>)+ </reunion>

Cu alte cuvinte, niciunul din termenii reuniți nu e el însuși o reuniune. Întotdeauna am cel puțin doi termeni reuniți.

<Concatenation-XML> ::= <concatenation> (<Reunion-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>) (<Reunion-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>)+ </concatenation>

După cum se poate observa, este foarte similară cu reuniunea.

<Kleene-Star-XML> ::= <star> (<Reunion-XML> | <Concatenation-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>) </star>

<Optional-XML> ::= <optional> (<Reunion-XML> | <Concatenation-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>) </optional>

<Plus-XML> ::= <plus> (<Reunion-XML> | <Concatenation-XML> | <Kleene-Star-XML> | <Optional-XML> | <Plus-XML> | <Basic-XML>) </plus>

<Basic-XML> ::= <basic> (<symbol-XML> | e | O) </basic>

Exemplu de expresie pentru $\{a, b\} : a^*$:

<RE><alphabet><symbol>a</symbol><symbol>b</symbol></alphabet> <star><basic><symbol>a </symbol></basic></star></RE>

4 Automate Finite Deterministe(AFD)

4.1 Specificații

În temă, un automat finit determinist este dat ca un tuplu, conform definiției:

<DFA> ::= (<states> , <alphabet> , <transitions> , <state> , (<states> | O))

<states> ::= { <state> (, <state>)* }

<state> ::= <name>

<name> ::= (<upper-case letter> | _) (<lower-case letter> | <digit> | _) *

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

<transitions> ::= { <transition> (, <transition>) * }

<transition> ::= d(<state> , <symbol>)= <state>

Pentru XML, tuplul e reprezentat în felul următor:

<DFA-XML> ::= <DFA> <states-XML> <alphabet-XML> <transitions-XML> <initial-XML> <final-XML> </DFA>

<states-XML> ::= <states> (<state-XML>)+ </states>

<state-XML> ::= <state> <state> </state>

<transitions-XML> ::= <delta>(<transition-XML>) + </delta>

<transition-XML> ::= <transition> <source> <state-XML> </source> <symbol-XML> <destination> <state-XML> </destination> </transition>

<initial-XML> ::= <initial> <state-XML> </initial>

<final-XML> ::= <final>(<state-XML>) * </final>

5 Automate Finite Nedeterministe(AFN)

5.1 Specificații

În temă, un automat finit nedeterminist este tot un tuplu, definit în felul următor:

<NFA> ::= (<states> , <alphabet> , <relations> , <state> , <states>)

<relations> ::= O | { <relation> (, <relation>) * }

<relation> ::= (<state> , <word> , <state>)

Pentru XML, tuplul e reprezentat în felul următor:

<NFA-XML> ::= <NFA> <states-XML> <alphabet-XML> <relations-XML> <initial-XML> <final-XML> </NFA>

<relations-XML> ::= <Delta>(<relation-XML>) * </Delta>

<relation-XML> ::= <relation> <source> <state-XML> </source> <word-XML> <destination> <state-XML> </destination> </relation>

6 Punctaj

Tema va fi testată cu un checker. Voi veți primi sau nu acel checker în funcție de cum e vremea pe 11 ianuarie. Probabil nu se va întâmpla.

6.1 Teste

Testele sunt publice și împărțite în două grupuri.

Primul grup, în valoare de 64 de puncte, conține cele 8 forme de reprezentare finită folosite pentru testarea temei 1. Acestea vor fi convertite din reprezentarea normală în XML și din XML în reprezentarea normală. Fiecare test va avea 4 puncte. (8 [reprezentări] * 2 [acțiuni] * 4 puncte = 64 puncte, testele 1-8)

Al doilea grup, în valoare de 36 de puncte, va conține trei reprezentări de dimensiune mai mare care trebuie convertite în și din XML, în valoare de 6 puncte fiecare. (3 [reprezentări] * 2 [acțiuni] * 6 puncte = 36 puncte, testele 9-11)

6.1.1 Tema 1.5

Pentru trimiterea temei 1.5, punctajul este de 200 de puncte. Primele 100 de puncte se dau pentru funcționalitatea temei 1, folosindu-se checker-ul de pe site. Următoarele 64 de puncte se dau pentru primul grup de teste ale temei 2.

Restul de 36 de puncte reprezintă 3 reprezentări finite convertite precum în tema 1 în toate celelalte forme de reprezentări finite în și din XML. Fiecare are 1 punct. (3 [reprezentari] * 3 [conversii de formă] * 2 [conversii de reprezentare] * 2 puncte = 36 puncte, testele 12-14)

Deadline hard: 20.01.2014, ora 03:00. Timp de lucru: zece zile lucrătoare.