Limbaje formale și tehnici de compilare

reprezentarea atomilor lexicali prin diagrame de tranziție

Definiții regulate

Expresiile regulate pot primi nume și astfel ele devin **definiții regulate**. Totodată, pentru ușurința scrierii, unele definiții pot interveni în formarea altor definiții. De exemplu, definiția unui caracter numeric (digit) „[0-9]" poate fi necesară în mai multe locuri și atunci poate fi inclusă în alte definiții. Dacă această definiție nu este folosită decât în interiorul altor definiții, fără a forma ea însăși un atom lexical, spunem despre ea că este un **fragment**. Unele unelte software care ajută la implementarea analizoarelor lexicale folosesc convenția ca atomii lexicali și fragmentele să înceapă cu o literă mare.

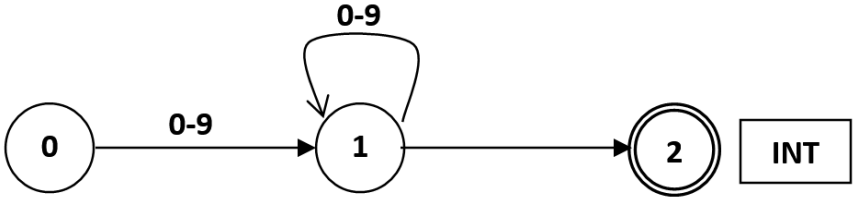
```
fragment DIGIT: [0-9] ;
INT: DIGIT+ ;
REAL: INT [.] INT ;
```

În exemplul de mai sus, DIGIT este un fragment lexical, deoarece nu este folosit decât ca parte constituentă a atomului INT. INT este atât atom lexical cât și parte componentă a definiției atomului REAL. Pentru definițiile regulate nu se admite recurența, deci o definiție nu se poate apela pe ea însăși, direct sau indirect. Astfel nu pot fi scrise cu ajutorul definițiilor regulate comentarii C „/*…*/" care să poată conține în interiorul lor alte comentarii C.
Mai există unele diferențe între definițiile regulate și expresiile regulate:

Definiții regulate (DR)	Expresii regulate (ER)
Spațiile nu sunt semnificative și DR poate fi formatată ca sa fie mai lizibilă	Orice spațiu este semnificativ și conteaza ca intrare
Deoarece o DR poate conține numele altor DR care intră în alcătuirea sa, cuvintele care se găsesc într-o DR au rol de identificatori, nu de caractere. Pentru a face diferența între acestea și caracterele obișnuite, prin conventie identificatorii pot fi ingrosati sau subliniati sau, în cazul formatelor text, orice caracter obișnuit să fie pus între apostroafe sau ghilimele.	Orice literă sau secvență de litere este tratată ca fiind caractere individuale, fără nicio altă semnificație.

Diagrame de tranziție (DT)

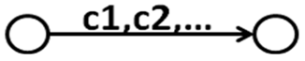
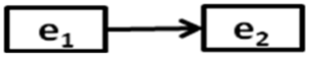
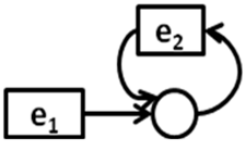
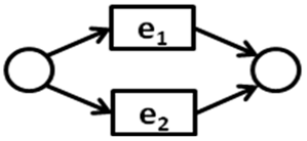
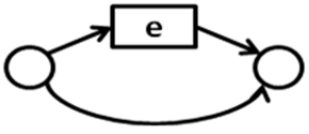
Diagramele de tranziție se bazează pe formalismul automatelor cu stări finite (prescurtat automate finite, AF). Ele pot fi folosite ca o modalitate de reprezentare a DR, pentru a se ușura implementarea acestora. DT se reprezintă grafic sub forma unui graf orientat care are ca noduri stările posibile în care se poate afla acea DT, iar ca arce tranzițiile pe care caracterele de intrare le declanșează de la starea curentă la o stare următoare.
Întotdeauna se pornește de la o stare inițială unică (0). Cand s-a reușit parcurgerea unei întregi DR, se ajunge într-o stare acceptoare sau finală (reprezentată cu 2 cercuri concentrice), ceea ce corespunde faptului că un nou atom lexical a fost format pe baza caracterelor de intrare. Acest ciclu se repetă până când nu mai sunt caractere de intrare.



În exemplul de mai sus este reprezentată o DT pentru DR „INT: [0-9]+”. Se poate constata că, începând din starea inițială 0, trebuie să se consume cel puțin un digit pentru a se ajunge în starea 1. În starea 1, dacă mai există și alți digiti în șirul de intrare, aceștia se consumă rămânând tot în starea 1. Tranziția care nu are nicio etichetă are semnificația de „altfel” sau „în caz contrar” și este analogică secvenței „else” de la „if”. Aceasta tranziție se testează întotdeauna ultima și ea nu trebuie să consume caracterul din șirul de intrare. De exemplu, dacă avem șirul „25*4”, caracterul „2” va fi consumat pe tranziția de la starea 0 la starea 1, iar caracterul „5” pe tranziția 1->1. Urmează „*” care constituie următorul atom lexical, deci s-a încheiat recunoașterea tuturor caracterelor care formează numărul „25”, iar cu tranziția 1->2 se va trece în starea finală 2, fără să se consume „*”. Dacă tranziția 1->2 ar fi consumat „*”, atunci acesta ar fi rămas consumat (fiindcă în această etapă s-ar fi returnat doar atomul „25”), iar la următoarea trecere prin DT s-ar fi recunoscut atomul „4”, deci „*” s-ar fi pierdut.

Reguli de construire a DT

Reprezentarea principalelor constituente ale expresiilor regulate în DT (cu pătrate s-au figurat expresii regulate care vor fi substituite ca atare):

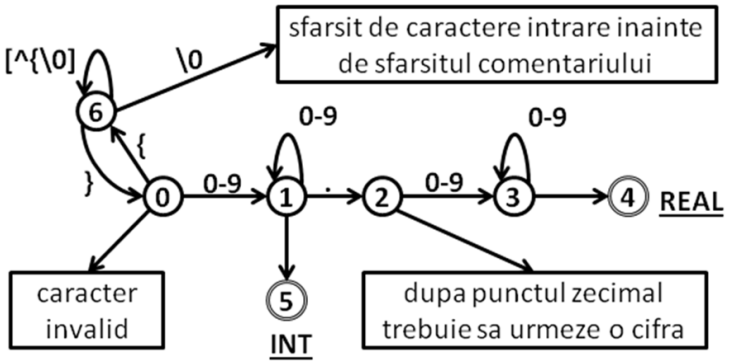
Expresie regulată	DT
caractere și clase de caractere – oricâte caractere sau clase de caractere care pleacă din aceeași stare și ajung într-o stare următoare comună, se pot reprezenta pe aceeași tranziție. Dacă se dorește negarea lor, se pot prefixa cu ^	
e*	
e_1e_2	
e+ - se folosește echivalența e+=ee*	
$e_1|e_2$	
e?	

Orice tranziție poate consuma cel mult un caracter (dar acesta poate fi luat dintr-o clasa de caractere). De exemplu, pentru a se reprezenta începutul comentariilor C de o singura linie „//”, trebuie reprezentate două tranziții, fiindcă sunt două caractere.

DR care nu trebuie reținute ca atomi lexicali pentru fazele următoare ale compilatorului (spații, comentarii), vor avea finalul tot în starea inițială. În acest fel, ele vor fi consumate fără a se genera atomi lexicali.

Dintr-o stare nu pot pleca tranziții care consumă aceleași caractere și nici mai multe tranziții „else”. În această situație, dacă ne-am afla în starea respectivă și ar trebui consumat un caracter pentru care sunt disponibile mai multe tranziții, nu am ști pe care să o urmăm. Aceste cazuri pot surveni când mai mulți atomi au același prefix, de exemplu INT și REAL încep ambii cu „[0-9]+”. În aceste situații se factorizează (se dă factor comun) prefixul comun.
În multe cazuri nu este explicitată o definiție pentru sfarsitul caracterelor de intrare (\0 sau EOF), dar un token final END este adăugat implicit.
Dacă două stări sunt legate doar printr-o stare „else”, atunci ele se pot unifica, păstrându-se toate tranzițiile care vin și pleacă din fiecare.
Dacă DT devine prea complicată, fiindcă trebuie adăugate multe DR, acestea se pot reprezenta separat, considerându-se starea inițială (0) ca fiind punctul lor comun. Și în acest caz trebuie păstrate proprietățile de mai sus. Fragmentele lexicale și identificatorii care apar în corpul DR trebuie expandati la constituentele lor (nu se vor figura ca nume/referințe în DT).
La fiecare stare finală (notată cu cercuri duble) se va nota atomul lexical corespunzător. Stările finale nu au tranziții care pleacă din ele.
Se pot prevedea stări și pentru posibile erori lexicale, de exemplu atunci când suntem într-o constantă șir de caractere (”…”) și apare sfârșitul de fișier înainte de ghilimeaua finală.

Exemplu: să se reprezinte DT pentru următoarele DR:
fragment DIGIT: [0-9] ;
INT: DIGIT+ ;
REAL: INT [.] INT ;
COMENTARIU: { [^}]* } ; //nu va genera un atom lexical



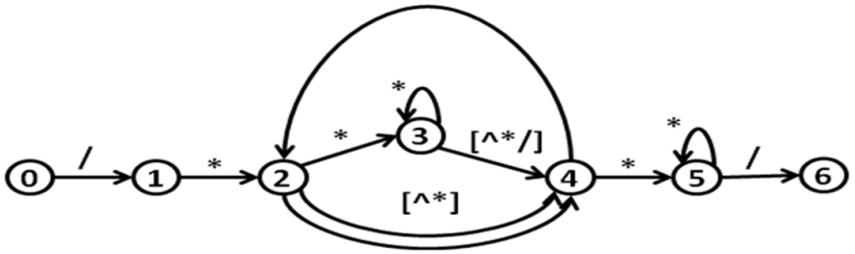
În exemplul de mai sus se pot constata următoarele:

- Toți identificatorii care apar în interiorul altor definiții (DIGIT, INT) au fost expandati la propriile lor expresii regulate. Deoarece DIGIT este doar un fragment, el nu formează atomi lexicali.
- Deoarece INT și REAL încep ambele cu „DIGIT+”, această secvență a fost factorizată și este un început comun pentru ambele DR. Dacă după „DIGIT+” urmează un punct (**.**), se vor considera numerele reale, altfel se obține un întreg.
- Secvența de stări corespunzătoare comentariilor (0->6->0) se încheie tot în starea inițială, deoarece comentariile nu trebuie sa genereze atomi lexicali.
- În anumite stări din DT nu sunt permise decât anumite caractere. De exemplu, după un punct zecimal (**.**) nu poate sa urmeze decât o cifră „[0-9]”. Dacă în această stare avem alt caracter de intrare, atunci se poate semnala o eroare. Erorile au fost reprezentate ca pătrate conținând un text explicativ.

Din cele prezentate mai sus, DT sunt automate finite nedeterministe (AFN), deoarece tranziția „else”, care nu consuma niciun caracter, corespunde unei ϵ-tranziții. Cu toate acestea, regulile de construcție și de structură a DT le fac sa poată fi implementate fără să existe nedeterminisme în privința tranziției de urmat într-o anumită stare cu un anumit caracter, ele comportandu-se astfel într-un mod determinist. De cele mai multe ori nedeterminismele se rezolvă simplu prin factorizarea prefixelor comune ale definițiilor regulate. Sunt însă situații în care în DR sunt nedeterminisme care sunt mai greu de rezolvat, de exemplu atunci când DR folosesc facilități care nu există în DT. Un astfel de exemplu este definiția comentariilor C:

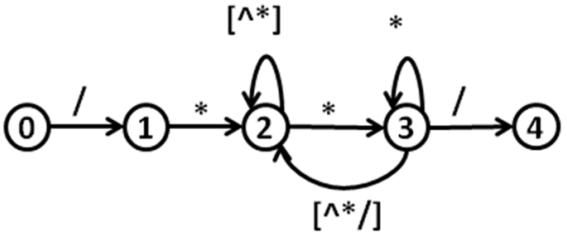
COMMENT: '/*' ([^*] | '*'+ [^*/])* '*'+ '/' ;

Regulile de construcție a DT din această DR conduc la:



Problema este dată de faptul că subexpresia '*'+ [^*/], care apare în stările 2->3->4, trebuie să fie parcursă doar dacă după consumarea ‘*’ nu urmează ‘/’. Dacă urmează ‘/’ după ‘*’, atunci trebuie să se revină înapoi din toate ‘*’ consumate și să se meargă pe tranziția else 2->4, urmând ca toate ‘*’ să fie consumate pe tranzițiile 4->5…->5. Această revenire nu este posibilă în DT, deoarece în DT nu se mai poate reveni dintr-un caracter deja consumat.

O alternativă ar fi să se încerce intuitiv construcția unei DT, pornind de la ce anume se dorește să se obțină, astfel încât să nu mai apară necesitatea revenirilor din caracterele consumate. Acest lucru este posibil deoarece în reprezentarea grafică se pot include tranziții care nu ar fi fost posibil să fie exprimate în DR, în special tranziții care revin la o stare anterioară consumând caractere. Rezultatul va fi de forma:



Pentru compilatorul AtomC este necesar să se figureze DT pentru atomii lexicali specificați în anexă, cu următoarele cerințe:

- cuvintele cheie nu se vor figura pe DT, ele urmând sa fie identificate ulterior pornind de la ID
- comentariile și spațiile nu formează atomi lexicali, ci doar se consumă