ПОДГОРОВКА ЗА ДЪРЖАВЕН ПИСМЕН ИЗПИТ ЗА СПЕЦИАЛНО КОМПЮТЪРНИ НАУКИ

CONTENTS

ЕАИ

ТИПОВЕ ЗАДАЧИ :

- Минимизиране на автомат
- Детерминиране на автомат
- Да се построи автомат от регулярен език

ПРАВИЛА ЗА ПОСТОЯВАНЕ НА АВТОМАТ ОР РЕГУЛЯРЕН ИЗРАЗ:

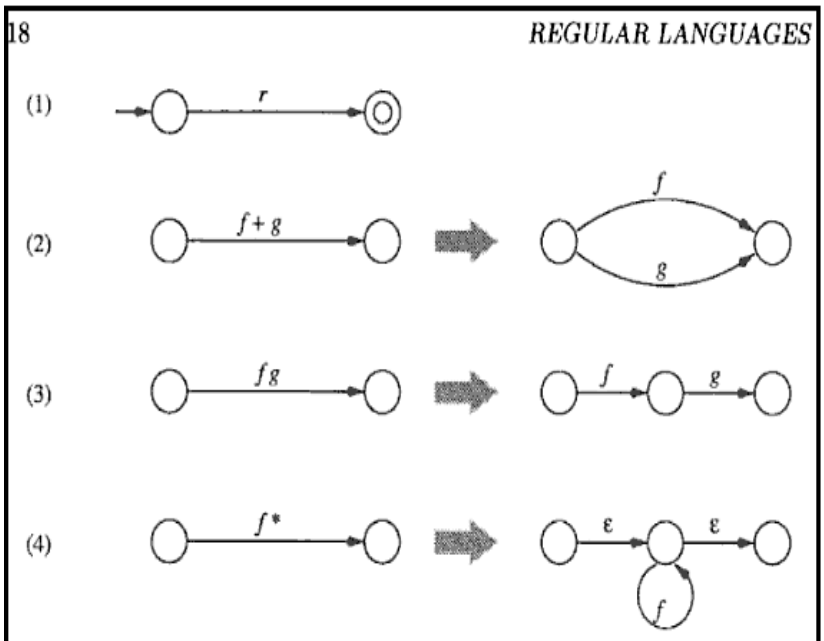

Figure 1.3: Graph $G(r)$ for regular expression r.

1.3 Graph Representations

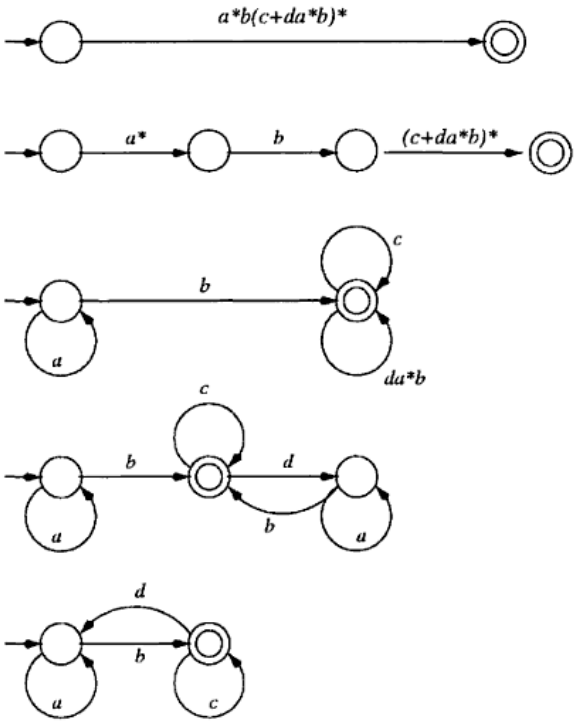

Figure 1.6: Labeled digraph $G(r)$ for $r = a^*b(c + da^*b)^*$.

РЕСУРСИ:

1. NFA and DFA from regular expression.Нагледно обяснение ☺
   http://www.scribd.com/doc/9234657/Construction-of-Nfa-and-Dfa-From-r
2. Book: Problem solving in automata, languages and complexity:

http://books.google.bg/books?id=VEHYzv0GHt8C&pg=PA53&lpg=PA53&dq=constructing+DFA+for++regular+expression&source=bl&ots=P7eFopWDeh&sig=D5WEZs0qUH41YKNP7sENFcWFgLk&hl=bg&ei=PCRRSoPnAqLimgOxx_S8BQ&sa=X&oi=book_result&ct=result&resnum=5#v=onepage&q=&f=false

Бележка: Това не ецялата книга ,но в частта,която е достъпна, има обяснение и примери за:

- Converting NFA to DFA (page 45)
- DFA and NFA from regular expression(page 18,19,21, пример: page 5

3. Минимизиране на автомат:
   - Минимизиране на автомат 1
   - Минимизиране на автомат 2

РЕКУРЕНТНИ ОТНОШЕНИЯ

Тук има много подробно обяснение как се решават : Рекурентни отношения

PROLOG

ОБЩИ БЕЛЕЖКИ:

В доста задачи се иска да се обясни какво ще бъде изведено, затова бъдето наясно как точно се генерира proof tree на Пролог.(виж ресурсите.) Дори да има задача за писане ,то тя няма да е сложна.

! – отрязва дървото наляво и надолу от себе си
Fail – атом, който никога не се удовлетворява

Lists – идеята при списъците е да се използва първият елемент от списъка (главата-Head) и след това рекурсивно да се повтори процеса с останалата част от списъка (Tail).

ПРИМЕР:

```
Member(H [H|T]).
Member(X, [H|T]:-Member(X,T).
```

Какво ще направи:

```
?-Member (X, [muscat, soho, martini])
```

Отговор: on backtracking ще генерира всички членове.

ПРИМЕР:

```
Int(0).
Int(X): - int(X1), X is X1 + 1.
```

Какво ще направи:

```
?-int(X).
```

Обяснение :
Първо се удовлетворява int(0). Тъй като искаме да го преудовлетворим се спускаме надолу по дървото => Достигаме до int(X), което от своя страна извиква int(X1), което се удовлетворява с 0 => X става 1(0 + 1) и т.н ...така се продължава и се генерират естествените числа.

РЕСУРСИ

1. Добро tutoral-че : http://www.mars-attacks.org/~boklm/prolog/page_5.html
2. Отсичане и генератори : http://moodle.openfmi.net/mod/resource/view.php?id=167
3. Много добро обяснение как се генерира proof tree и като цяло обяснение на Execution model-a v Prolog.: http://books.google.bg/books?id=noJW-_zXKskC&printsec=frontcover&dq=proof+trees+Prolog&source=gbs_similarbooks_s&cad=1#v=onepage&q=&f=false
   Бележки : Погледни Глава 4:Execution Model in Prolog (page 20).Към края на главата има и информация за cut i fail.

ПРИМЕРНИ ЗАДАЧИ:

Числа на фибоначи,факториел, дали число е между други 2 числа, минимален елемент на списък revrse na списък

Решения:

Фибоначи:

```
fib(0,1).
fib(X1,X):-fib(P1,X1),X is P1+X1.
fib(N,X):-fib(N,P1,X).


fib(X):-fib(_,X).

#--------------------------------
```

Факториел:

```
fib(0,1).
fib(X1,X):-fib(P1,X1),X is P1+X1.
fib(N,X):-fib(N,P1,X).

fib(X):-fib(_,X).

#--------------------------------
```

Дали число е между 2 числа:

```
between(A,A,B).
between(X,A,B):-C is A+1,A<B,between(X,C,B).
#------------------------------
```

Минимален елемент:

```
min(X,[X])
min(X, [Z|T]):- min(Y,T), m(X,Z,Y).
```

```
m(B,B,C):- B < C.

m(C,B,C):- C < B.

min(X,L):- member(X,L),not(member(Y,L)),Y<L.

#---------------------------------
```

Reverse na списък:

```
reversse([],[]).
reverse([H|T],Y).

reverse(T,X).
```

ПРИНЦИП НА РЕЗОЛЮЦИЯТА

ОБЩИ БЕЛЕЖКИ:

Основни преобразувания :

1. ~ (~A) = A

2. ~ (A&B) = ~A v ~B

3. ~ (A v B) = ~A & ~B

4. A&(BvC) = A&B v A&C

5. A v (B&C) = (A v B) & (B v C)

6. A ⇔ B = (A => B) & (B =>C)

7. A => B = ~A v ~B

8. ~ (∀ X(A)) = ∃X(~A)

9. ~ (∃ X(A)) = ∀X(~A)

10. ∀ X(A) = ∀Y (A(Y))

11. ∃ X(A) = ∃Y(A(Y))

Допълнителни задачи

РЕСУРСИ

1. Тук има много подробно обяснение как се решават : Принцип на резолюцията
2. Учебника по Логическо програмиране на Магдалина Тодорова

SCHEME

ОБЩИ БЕЛЕЖКИ:

Car cdr caadr caddr ….

Какво ще изведе:

(cadr (list '(() (16 (0))) (caadr '(((14 10) 2) (1 (11)) ((8) (9))))))

Разлика между let и let*

Let – свързването на променливите е псевдопаралелно

Let* - свързването на променливите е последователно

Reminder vs quotient

(reminder m n)– остатъка при деление на 2 числа

(quotient m n) – цялата част при деление на 2 m и n

map and apply

apply – прилага операция към елементите на списък

пример: (apply + `(1 2 3)) → 6

map : прилага процедура към всеки от елементите на списък и връща списък с ножите елемент

(map (lambda (x) (* x x)) `(1 2 3))) → (1 4 9)

append vs cons

(append `(1((2 3))) '(1)) → (1 ((2 3)) 1)

(cons `(1((2 3))) '(1)) → ((1 ((2 3))) 1)

eq? vs eqv?

1) Предикати за проверка за равенство

(eq? s1 s2**)** ⟶ { ***#t***, ако **[s1]** и **[s2]** са идентични (ако **[s1]** и **[s2]** са означения на един и същ обект, т.е. на един и същ участък от паметта);
#f, в противен случай.

Забележки:

1. Обикновено предикатът ***eq?*** се зползва за проверка на това, дали **[s1]** и **[s2]** са еднакви символни атоми (има и други случаи, в които ***eq?*** дава резултат ***#t***, но те са много малко).

2. За сравняване на числа е добре да се използва аритметичният предикат **=** или предикатът ***eqv?*** (той обединява в определен смисъл действието на ***eq?*** и **=**).

{ ***#t***, ако **[s1]** и **[s2]** са еквивалентни S-изрази (в частност, списъци),

т.е.или са еднакви символни атоми, или са еднакви низове,

(equal? s1 s2**)** ——> или са равни числа от един и същ тип, или са точкови

двойки с еквивалентни ***car*** и ***cdr*** части;

#f, в противния случай.

Рекурсивен и итеративен стил.Сравнете долните функции за намиране дължината на списък:

- в рекурсивен стил

```
(define (length l)
  (if (null? l)
                0
                (+ 1 (length (cdr l)))))
```

- в итеративен стил

```
(define (length l)
     (define (length-iter arg count)
                (if (null? arg)
                     count
                   (length-iter (cdr arg) (+ 1 count))))
      (length-iter l 0))
```

За да се използват някои от вградените функции за списъци трябжа да се добави библиотеката compat.ss(това на изпита няма да трябва, но е полезно да се знае докато се упражнявате).Може да я добавите така:

```
 (require (lib "compat.ss"))
```

Задачи : flat, accumulate, reverse ,deep reverse,

Задачи за граф(представянето най често е с асоциативни списъци) : обхождане в ширина и дълбочина.

Ресурси:

1. Тема 19 от държавния :
   Основни понятия и функции
   Списъци
2. Учебника на Нишева по функционално програмиране
3. Упражненията в moodle : Упражнения по функционално програмиране

BASH SCRIPT

ОБЩИ БЕЛЕЖКИ:

ПРИМЕР: (обяснение на set $var)

```
for var in a1 a2 a3
do
   set $var
done
```

set $var присвоява стойност на първия позиционен параметър => в горния случай след края на цикъла $1 има стойност a3.

ЗАДЪЛЖИТЕЛНИ КОМАНДИ:

`$#` Брой аргументи, предадени на shell без нулевия.
`$*, $@` Всички аргументи, предадени на shell без нулевия, разделени с интервал.
`$?` Код на завършване на последната изпълнена команда в привилегирован режим.
`$$` Идентификатор (pid) на процеса shell.
`$!` Идентификатор (pid) на последния изпълнен фонов процес. Бази данни

ИЗПОЛЗВАНИ КОМАНДИ:

shift
ls
grep
who
wc
cat
cd
pwd
,chgrp
read
echo
kill
rm
cp
diff
tee

ПРИМЕРНИ ЗАДАЧИ:

```
for var in a1 a2 a3
do
      set $var
done
shift
listpar=` echo $* `
if [ -n "$listpar" ]
      then
      true
else
      false
fi
echo $?
```

```
echo $listpar

#-------------------------------------------------------------
```

abc
copy.c
copy.out
opit.out
wcount.c
xyz
каква стойност ще изведе на стандартния изход следната командната процедура:

```
cd /usr
a=` ls *.out | wc –l `
if [ $a -gt 2 ]
then a=2
else echo $a
fi
echo $a

#-------------------------------------------------------------------
```

Ако съдържанието на директория /home е
dum.exe
err1
gref.out
mnn.exe
opit.exe
st1
us1.out
каква стойност ще изведе на стандартния изход командната процедура
(верният отговор да се огради с кръгче)

```
br=2
cd /home
for var in *.exe
do
echo $var >> fnam
br=` expr $br + 1 `
done
br=`grep out fnam | wc –l `
echo $br
```

РЕСУРСИ

1. Лекция по ОС(добре е цялата да се знае) : ПРОГРАМИРАНЕ НА КОМАНДЕН ЕЗИК В UNIX И LINUX

ООП - NOT FINISHED

Задача за реализиране на йерархия от класова

Пример :

Особености :

Задача от тип : какво ще бъде изведено

Виртуално наследяване

Вуртулани методи

Статично и динамично свързване

СТРУКТУРИ ОТ ДАННИ – NOT FINISHED

In progress…

СИСТЕМНО ПРОГРАМИРАНЕ – NOT FINISHED

Пример :

Fork

Exec и всички от групата

Особеност при exec: сменя образа на процеса

БАЗИ ДАННИ - NOT FINISHED

Задачи върху select

Особености:

агрегатни функции

Важно при having + агрегатни функции

ДЪРЖАВНИ ИЗПИТИ , ДАВАНИ МИНАЛИ ГОДИНИ:

На компютърни науки: Exams\KN

http://www.fmi.uni-sofia.bg/exams/drzhaven-izpit/temi-drzhaven-izpit/temi-za-specialnost-kompyutrni-nauki

На инфоматика : Exams\Informatika

http://www.fmi.uni-sofia.bg/exams/drzhaven-izpit/temi-drzhaven-izpit/temi-za-specialnost-informatika

Решения:

Това са решения на някои от колегите на задачите давани от минали години:
http://docs.google.com/Doc?id=df5m6f3k_7d2kkqzdv