Inteligenţă Artificială
Temă de casa

May 29, 2017

Titlu: Top-down proof with SLD resolution

Profesor: Costin Bădică

Student: Voiculescu Ioan-Valentin

Facultate: Automatică, Calculatoare şi Electronică

Anul: II

Specializarea: Calculatoare Română

Grupa: CR 2.2 B

Contents

Abstract

Acest document prezinta obiectivele si metodologia pentru dezvoltarea temelor de Inteligenta Artificiala. Top-Down Proof Procedure este o metoda alternativa de verificare a cautarii inapoi sau de sus in jos dintr-o interogare pentru a determina daca aceasta este o consecința logica a clauzelor definite.

1 Declaratia problemei

1.1 Titlu

Top-down proof with SLD resolutions.

1.2 Introducere

Această procedură se numește rezoluție clauză propozitivă definită sau rezoluție SLD, în care SL reprezintă Selectarea unui atom folosind o strategie liniară, iar D reprezintă clauze definite. Este o instanță a metodei de rezoluție mai generală.

Procedura de verificare de sus în jos poate fi înțeleasă în termenii clauzelor de răspuns. O clauză de răspuns este de forma:

$$yes \leftarrow a_1 \wedge a_2 \wedge ... \wedge a_m$$

Unde yes este un atom special. Intuitiv, va fi adevărat atunci când răspunsul la interogare este "yes".

Având o clauză de răspuns, algoritmul de sus în jos selectează un atom din corpul clauzei de răspuns. Să presupunem că selectează a_i. Atmosul selectat este numit scop. Algoritmul continuă prin pașii de rezoluție. Într-o etapă a rezoluției, alege o clauză definită în KB cu a_i ca cap. Dacă nu există o astfel de clauză, ea nu reușește.

O derivare SLD a unei interogări $ask\ q_1 \wedge ... \wedge q_k$ din baza de cunoștințe KB este o secvență de clauze de răspuns $y_0, y_1, ..., y_n$ astfel încât:

- y_0 este clauza de răspuns corespunzătoare interogării inițiale, și anume clauza de răspuns $yes \leftarrow q_1 \wedge ... \wedge q_k$;
- y_i este resolventul y_{i-1} cu o clauză definită în KB;
- y_n este un răspuns.

2 Pseudocod al algoritmilor

O modalitate de a gândi despre algoritm este că algoritmul de sus în jos menține o colecție G de atomi ca să demonstreze. Fiecare atom care trebuie dovedit este un scop. Inițial, G este setul de atomi din interogare. O clauză $a \leftarrow b_1 \wedge ..... \wedge b_p$ înseamnă că obiectivul a poate fi înlocuit de obiectivele $b_1, ..., b_p$. Fiecare b_i este un subgol al lui a. G care trebuie demonstrat corespunde clauzei de răspuns $yes \leftarrow G$.

O strategie comună de selecție este aceea de a selecta atomul din stânga în ordonarea atomilor. Aceasta nu este o strategie corectă; Este posibil ca aceasta să intre într-o buclă infinită, atunci când o strategie diferită ar eșua. Cea mai bună strategie de selecție este selectarea atomului care este cel mai probabil să eșueze.

Algoritmul nedeterminist de sus în jos împreună cu o strategie de selecție induc un grafic de căutare, care este un arbore. Fiecare nod din graficul de căutare reprezintă o clauză de răspuns. Vecinii unui nod $yes \leftarrow a_1 \wedge ... \wedge a_m$, unde a_i este atomul selectat, reprezintă toate clauzele de răspuns posibil obținute prin rezolvarea pe a_i. Există un vecin pentru fiecare clauză definită al cărei cap este a_i. Nodurile țintă ale căutării sunt de forma $yes \leftarrow$.

Graficul de căutare pentru o derivare SLD, presupunând că atomul din stânga este selectat în fiecare clauză de răspuns, este prezentat în Figura 1. Având în vedere baza de cunoștințe:

$a \leftarrow b \wedge c.$
$a \leftarrow g.$
$a \leftarrow h.$
$b \leftarrow j.$
$b \leftarrow k.$
$d \leftarrow m.$
$d \leftarrow p.$
$f \leftarrow m.$
$f \leftarrow p.$
$g \leftarrow m.$
$g \leftarrow f.$
$k \leftarrow m.$
$h \leftarrow m.$
$p.$

Și interogarea:

$ask\ a \leftarrow c \wedge d.$

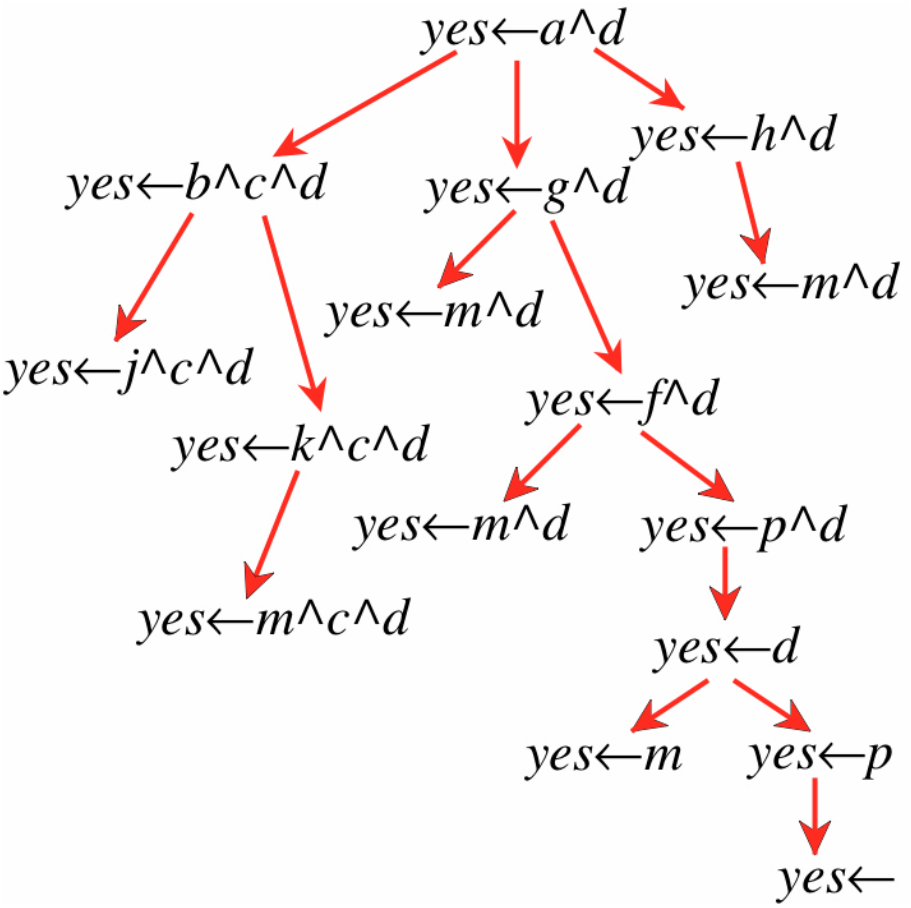

Figure 1: Un grafic de căutare pentru o derivare de sus în jos.

2.1 FUNCTIA citire

```
1. v[0] ←0
2. while(!feof(file))
3.     c ←fgetc(file)
4.    if(((c≥48)and (c≤57)) or ((c≥97) and (c≤122)))then
5.        d[strlen(d)] ←c
6.        d[strlen(d)] ←NULL
7.    else
8.            if(strlen(d)!=0) then
9.                if((d[0]≥97) and (d[0]≤122))then
10.                    v[++v[0]]=d[0]
11.                if((d[0]≥48) and (d[0]≤57))then
12.                    v[++v[0]]=atoi(d)
13.                for i ←0, n, 1 execute
14.                    d[i] ←NULL
```

2.2 FUNCTIA atribuire

```
1. p ←v+1
2. n ←n*p
3. p ←p+1
4. for i ←0, n, 1 execute
5.    baza[i].a ←*p
6.    p ←p+1
7.    baza[i].a ←*p
8.    p ←p+1
9.    for j ←0,baza[i].k, 1 execute
10.       baza[i].r[j][0] ←*p
11.       p ←p+1
12.           for m ←1,baza[i].k
13.               baza[i].r[j][m] ←*p, 1 execute
14.               p ←p+1
15. for i ←1,head → r[0], 1 execute
16.    head → r[i] ←p
17.    p ←p+1
```

2.3 FUNCTIA creare-arbore

```
1.if (p! ←NULL) then
2.    for i, sw ←0, (i¡n) and (sw==0), 1 execute
3.       if(r[1]==baza[i].a) then
4.           sw ←1
5.           id ←1
```

```
6.    if(sw!=0) then
7.        if (baza[id].k >0) then
8.            for i ←0, baza[id].k, 1 execute
9.                for j ←1,r[0], 1 execute
10.                   rr[j] ←r[j]
11.               rr[0] ←r[0]
12.               for h ←1,baza[id].r[i][0], 1 execute
13.                   for j ←rr[0],1,-1 execute
14.                       rr[j + 1] ←[j]
15.                   rr[0] ←rr[0]+1
16.               for h ←1,baza[id].r[i][0], 1 execute
17.                       rr[h] ←baza[id].r[i][h]
18.               p →leg[i] ←new Node
19.               p →leg[i] →r[0] →rr[0]
20.               for l ←1,rr[0], 1 execute
21.                   p →leg[i] →r[l] ←rr[l]
22.               reading(p →leg[i], rr)
23.           p →leg[baza[id].k] ←NULL
24.       else
25.           if(baza[id].k ←0) then
26.               if(p →r[0]>0) then
27.                   forj ←1, r[0], 1 execute
28.                       r[j] ←r[j+1]
29.                   r[0] ←r[0]-1
30.                   p →leg[0] ←new Node
31.                   p →leg[0] →r[0] ←r[0]
32.                   forl ←1, r[0], 1 execute
33.                       p →leg[0] →r[l] ←r[l]
34.                   p →leg[1] ←NULL
35.                   reading(p →leg[0],r)
36.               else
37.                   p →leg[0] ←NULL
38.   else
39.       p →leg[0] ←NULL
```

2.4 FUNCTIA de afisare

```
1. if(p! ←NULL) then
2.    if(p →r[0] ←0) then
3.        printf("0 ")
4.    fori ←1,p →r[0], 1 execute
5.        printf("c",p →r[i])
6.    printf("b")
7.    if(q! ←NULL) then
8.        printf("(")
```

9. **for**$j \leftarrow$1,$q \rightarrow$r[0], 1 execute
10. printf("b")
11. $q \leftarrow$p
12. **else**
13. $q \leftarrow$p
14. **for**$i \leftarrow p \rightarrow leg[i]! \leftarrow$NULL, 1 execute
15. display($p \rightarrow$leg[i], q)

3 Numarul Problemei

For each letter in my name I searched for the ASCII equivalent:

1. $V \leftarrow 86$
2. $o \leftarrow 111$
3. $i \leftarrow 105$
4. $c \leftarrow 99$
5. $u \leftarrow 117$
6. $l \leftarrow 108$
7. $e \leftarrow 101$
8. $s \leftarrow 115$
9. $c \leftarrow 99$
10. $u \leftarrow 117$
11. $I \leftarrow 73$
12. $o \leftarrow 111$
13. $a \leftarrow 97$
14. $n \leftarrow 110$
15. $Suma\ este \leftarrow 1449$
16. $Problema\ este \leftarrow 2$

4 Proiectarea Aplicației

4.0.1 Input si Output

Datele de intrare si de iesire se gasesc in fisierul in.txt respectiv fiserul out.txt

4.0.2 Compatibilitate

Tema de casa a fost realizata in sitemul de operare Windows. Compilatorul folosit este GNU GCC Compiler cu standardul C99.

References

[1] http://artint.info/html/ArtInt_110.html.