Compiladors: Examen de teoria
26 de juny de 2014, 8:00

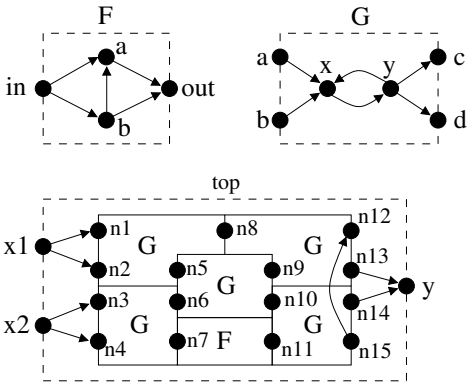
Duració de l'examen: 2.5 hores

Publicació de les notes: 27 de juny (tarda)
Revisió de notes: 30 de juny (11:00-12:00)

1 Disseny de gramàtica (2.5 punts)

Dissenyar una gramàtica per a descriure grafs jeràrquics, de manera semblant a com es mostra a la figura. Suposarem que l'analitzador lèxic ja ha reconegut els tokens per a les paraules clau (`GRAPH`, `ENDGRAPH`) i pels identificadors (`ID`). Els altres tokens s'hauran d'especificar a la pròpia gramàtica, per ex. `'('`, `'>'`, etc. La gramàtica ha de tenir S com a símbol inicial. Suposarem que:

- Cada entrada ha de tenir com a mínim un graf.
- Un graf pot tenir un conjunt buit de paràmetres, per exemple `G()`.
- El cos d'un graf pot ser buit, per exemple: `graph G(m, n) endgraph`.




```
graph F(in, out)
  in > a > out;
  in > b > a;
  b > out;
endgraph

graph G(a, b, c, d)
  a > x > y > x;
  b > x; y > c; y > d;
endgraph

graph top(x1, x2, y)
  x1 > n1; x1 > n2;
  G(n1, n2, n8, n5);
  G(n3, n4, n6, n7);
  x2 > n3; x2 > n4;
  F(n7, n11);
  G(n5, n6, n9, n10);
  G(n8, n9, n12, n13);
  G(n10, n11, n14, n15);
  n13 > y; n14 > y;
  n15 > n12;
endgraph
```

Resposta:

$$
\begin{array}{rcl}
S & \rightarrow & G+ \\
G & \rightarrow & \texttt{GRAPH}\ \mathit{Header}\ \mathit{Body}\ \texttt{ENDGRAPH} \\
\mathit{Header} & \rightarrow & \texttt{ID}\ \texttt{'('}\ \mathit{Params}?\ \texttt{')'} \\
\mathit{Params} & \rightarrow & \texttt{ID}\ (\texttt{','}\ \texttt{ID})* \\
\mathit{Body} & \rightarrow & ((\mathit{Edges} \mid \mathit{Header})\texttt{';'})* \\
\mathit{Edges} & \rightarrow & \texttt{ID}\ (\texttt{'>'}\ \texttt{ID})+
\end{array}
$$

2 Anàlisi sintàctica (2.5 punts)

Donada la següent gramàtica:

$$
\begin{array}{lrcl}
(1) & S & \rightarrow & A\$ \\
(2) & A & \rightarrow & AaB \\
(3) & A & \rightarrow & Bb \\
(4) & A & \rightarrow & b \\
(5) & B & \rightarrow & cBd \\
(6) & B & \rightarrow &
\end{array}
$$

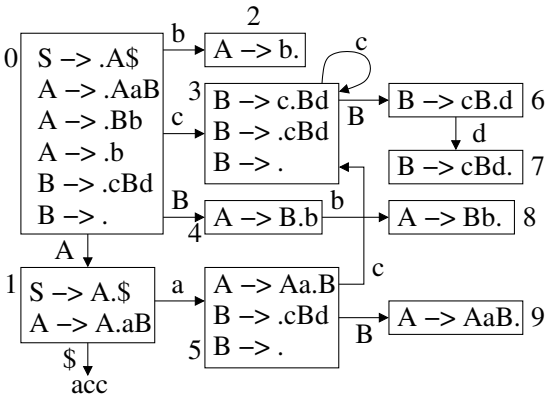
- Calcular *First* i *Follow* pels símbols A i B.

  Resposta:

	First	*Follow*
A	$\{b, c\}$	$\{a, \$\}$
B	$\{c, \varepsilon\}$	$\{a, b, d, \$\}$

- Generar l'autòmat i la taula LR(1).

  Resposta:



	a	b	c	d	$	A	B
0	r6	s2/r6	s3	r6	r6	1	4
1	s5				acc		
2	r4				r4		
3	r6	r6	s3	r6	r6		6
4		s8					
5	r6	r6	s3	r6	r6		9
6				s7			
7	r5	r5		r5	r5		
8	r3				r3		
9	r2				r2		

Existeix un conflicte *shift/reduce* a l'estat 0 amb entrada b.

3 Generació de codi (2.5 punts)

Generar el codi intermig corresponent al següent fragment de codi C++ fent servir el mètode d'avaluació d'expressions booleanes basat en "backpatching". Cal escriure el resultat després de fer optimització de codi. Per a generar codi es poden fer les següents suposicions:

- Les constants `false` i `true` és poden representar amb els enters 0 i 1, respectivament.
- Totes les variables que apareixen al codi són locals.
- El vector `a` és un array d'enters de 4 bytes.

```
i = 0;
found = false;
while (not (found or not (i < n))) {
  if (a[i] != 0 and a[i] == x) found = true;
  else i = i + 1;
}
```

Resposta:

```
    i = 0
    found = 0
L1: if found goto L3
    if i >= n goto L3
    t1 = i*4
    t2 = a[t1]
    if t2 = 0 goto L2   // ifFalse t2 != 0 goto L2
    if t2 != x goto L2  // ifFalse t2 == x goto L2
    found = true
    goto L1
L2: i = i + 1
    goto L1
L3:
```

4 Optimització de codi (2.5 punts)

Considerar el següent codi:

```
        i = 0
        a = n*3
        if i >= a goto L2
    L1: b = i*4
        c = p+b
        d = M[c]
        e = d*2
        f = i*4
        g = p+f
        M[g] = e
        i = i+1
        a = n*3
        if i < a goto L1
    L2:
```

Preguntes:

- Suposant que només la variable `M` està viva al final del codi, quines són les variables vives abans d'executar-se la instrucció de `L1`?. I després de la instrucció `g = p+f`?
- Optimitzar el codi tant com com sigui possible utilitzant les optimitzacions que s'han vist durant el curs. Cal explicar cadascuna de les optimitzacions realitzades i donar el codi final després d'optimitzar. Per a facilitar l'explicació es pot donar la evolució del codi progressivament després de cada subconjunt d'optimitzacions.

Resposta: Les variables vives a `L1` són `M`, `i`, `n` i `p`. Les variables vives després de `g = p+f` són `M`, `e`, `g`, `i`, `n` i `p`. El codi optimitzat podria ser el següent, després de detectar invariants (`n*3`), expressions comunes (`n*3`, `p+b`), propagar còpies i eliminar codi mort.

```
    i = 0
    a = n*3
    if i >= a goto L2
    a = n*3
L1: b = i*4
    c = p+b
    d = M[c]
    e = d*2
    f = b
    g = p+b
    M[g] = e
    i = i+1
    if i < a goto L1
L2:
```

```
    i = 0
    a = n*3
    if i >= a goto L2
    a = a
L1: b = i*4
    c = p+b
    d = M[c]
    e = d*2
    g = c
    M[g] = e
    i = i+1
    if i < a goto L1
L2:
```

```
    i = 0
    a = n*3
    if i >= a goto L2
L1: b = i*4
    c = p+b
    d = M[c]
    e = d*2
    M[c] = e
    i = i+1
    if i < a goto L1
L2:
```