Control LP (Compiladors): Un llenguatge per la gestió de models de processos acíclics

Cal fer un compilador per interpretar un llenguatge simple de definició i anàlisi de models de processos. Un model de processos consta d'un conjunt de rols, on cada rol descriu un subprocés a executar per aquell rol. Un subprocés es una descripció d'una tasca complexa sobre un conjunt finit d'activitats. En concret, es fan servir quatre operadors sobre les activitats: sequencia (;), paral·lelisme (+), exclusió (#) i inclusió (|). A més de la definició de rols, activitats de diferents rols poden estar relacionades via una connexió directa. Addicionalment, es poden definir fitxers compartits entre varis rols, a partir de lectures/escriptures desde les activitats. Finalment, un cop definit el procés, es pot analitzar mitjançant un conjunt de preguntes sobre els camins crítics, diferències entre dos rols i correctessa en el ús d'un fitxer. A continuació podeu veure un exemple complet d'aquest llenguatge.

```
start                                   // definicio del rol Sales. En paral·lel s'executa
A;B;C|D+E|F#G                           // (branca 1) o be (A, despres B, C), o be D, o be tot
end                                     // (branca 2) o be E o F (o els dos), i G
Sales

start
X+Y;Z#M|P+Q;R;T
end
Accounting                              // definicio del rol Accounting.

start
T1;(T2+T3;(T4+T5)#T6)#T7|T8
end
HumanResources                          // definicio del rol HumanResources.

connection A X                          // connexio entre les activitats A (Sales) i X (Accounting)
connection D M
file F -> B                             // l'activitat B llegeix del fitxer F
file F <- M                             // l'activitat M escriu al fitxer F
file Y -> A

QUERIES

critical Sales                          // cami critic a Sales es MAX(MAX(3,1),MAX(MAX(1,1),1)) = 3
critical Pepito                         // error rol Pepito no existeix
difference Sales Accounting             // Cert: rol Sales es diferent a Accounting
difference Sales Sales                  // Fals
correctfile F                           // Cert: hi ha almeny un rol que escriu i un que llegeix a F
correctfile Y                           // Fals: cap rol escriu a Y
```

Fixeu-vos en la diferència entre la semàntica dels operadors: mentre que l'operador "|" pot executar una o més branques (en paral·lel), l'operador "+" les executa totes en paral·lel. Finalement l'operador "#" només executarà una de les branques. Si mireu una part del model (a l'anvers de la pàgina) construit a partir de l'exemple, podreu observar que l'operador "+" es menys prioritari que "#", que a la vegada es menys prioritari que "|", que igualment es menys prioritari que ";". Si es vol canviar la prioritat cal fer servir els parèntesis (mira el rol HumanResources). L'arbre que hi ha al final de l'enunciat detalla de forma explícita les prioritats i precedències.

[Part 1: 50% nota] Defineix la part lèxica i sintàctica. Fès la gramàtica per a que PCCTS pugui reconèixer-la i decorar-la per generar l'AST mostrat a l'anvers de la pàgina.

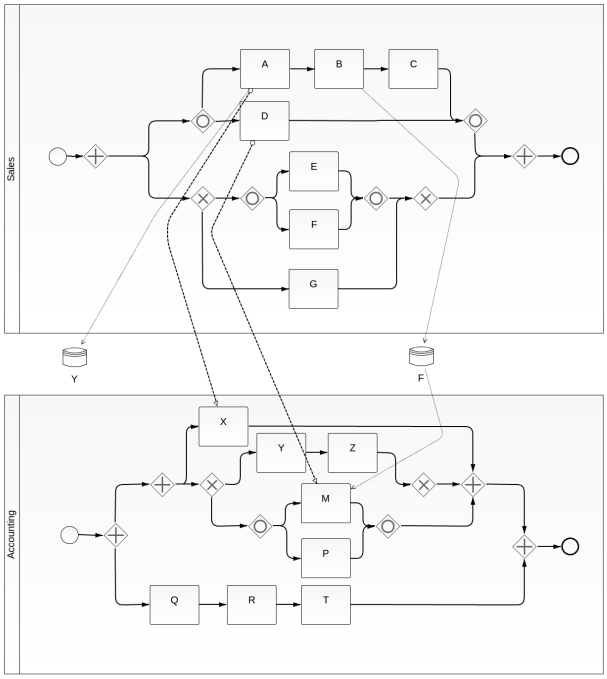


Figura 1: Part del model relacionat amb els rols Sales i Accounting.

[Part 2: 50% nota] Interpretació: fés els mètodes

```
int critical(AST *a, string role)
bool difference (AST *a, string role1, string role2)
```

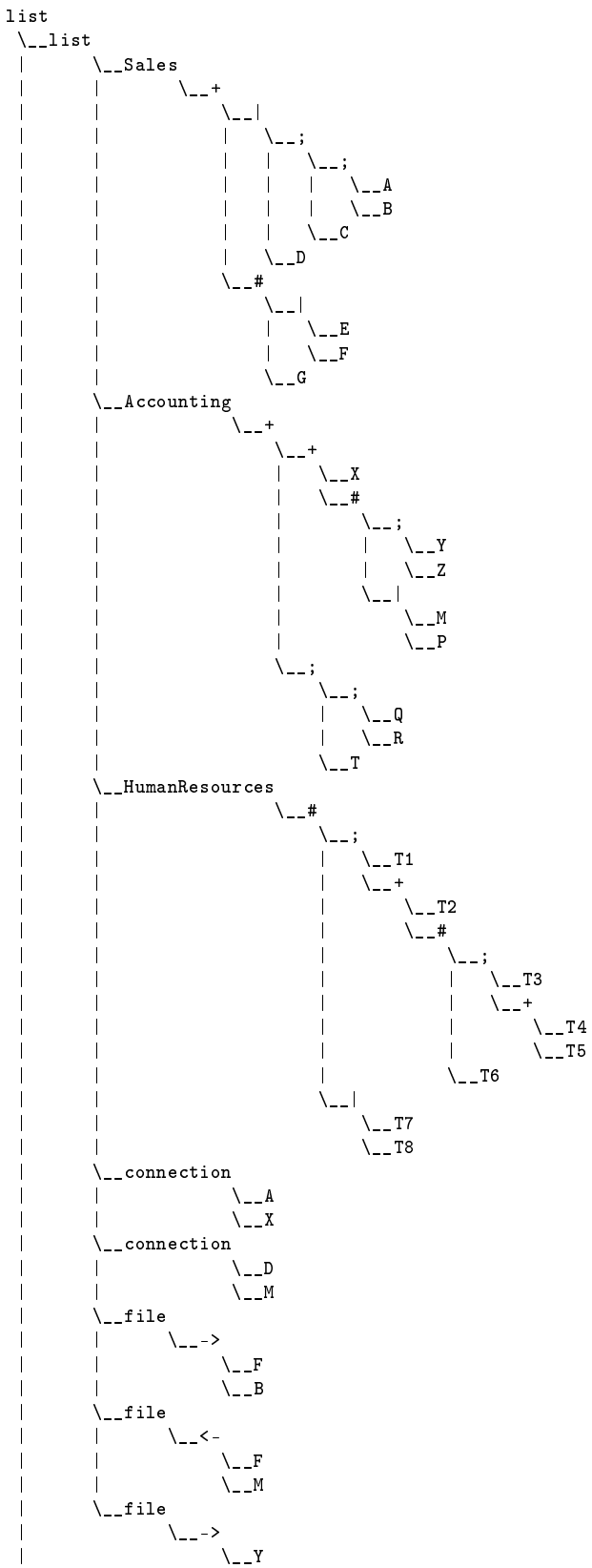
Que es cridaran al trobar una instrucció de critical o difference. La funcio critical ha de calcular la mida (en nombre d'activitats) del camí més llarg en el graf subjacent. Per calcular-la penseu que quan dues branques son paral·leles, inclusives o exclusives, el camí critic serà el màxim dels camins crítics de les dues branques. Per tant, aquests tres operadors tenen el mateix tractament en el càlcul dels camins crítics. En canvi, si hi ha dues branques seqüencials, el camí crític sera la suma de les dues. Podeu veure l'exemple del rol Sales per entendre el càlcul del camí crític.

La funció difference determinarà si dos arbres són el mateix, és a dir, representen la mateixa estructura d'arbre (oblidant les activitats, centrant-se només en els operadors). Per a calcular-ho, simplement cal recòrrer els dos arbres per nivells buscant alguna diferència. Per exemple, Sales i Accounting són diferents perque al segón nivell Sales té una inclusió mentre que Accounting té un paral·lelisme. En cas de trobar-la, o que un recorregut acabi mentre l'altre no, es confirmarà la diferència. Altrament els arbres són el mateix. Noteu que no cal considerar la commutativitat dels operadors, per tant si dos arbres tenen en un nivell els mateixos operadors però definits en diferent ordre seran considerats diferents.

En el codi que has de fer, assumeix que ja hi ha definida la funció:

```
AST *findRole(string id)
```

que donat el nom d'un rol, retorna el node de l'AST on està definit.


list
__list
__Sales
__+
__|
__;
__;
__A
__B
__C
__D
__#
__|
__E
__F
__G
__Accounting
__+
__+
__X
__#
__;
__Y
__Z
__|
__M
__P
__;
__;
__Q
__R
__T
__HumanResources
__#
__;
__T1
__+
__T2
__#
__;
__T3
__+
__T4
__T5
__T6
__|
__T7
__T8
__connection
__A
__X
__connection
__D
__M
__file
__->
__F
__B
__file
__<-
__F
__M
__file
__->
__Y

```
|                  \__A
\__list
       \__critical
       |          \__Sales
       \__critical
       |          \__Pepito
       \__difference
       |            \__Sales
       |            \__Accounting
       \__difference
       |            \__Sales
       |            \__Sales
       \__correctfile
       |             \__F
       \__correctfile
                     \__Y
```