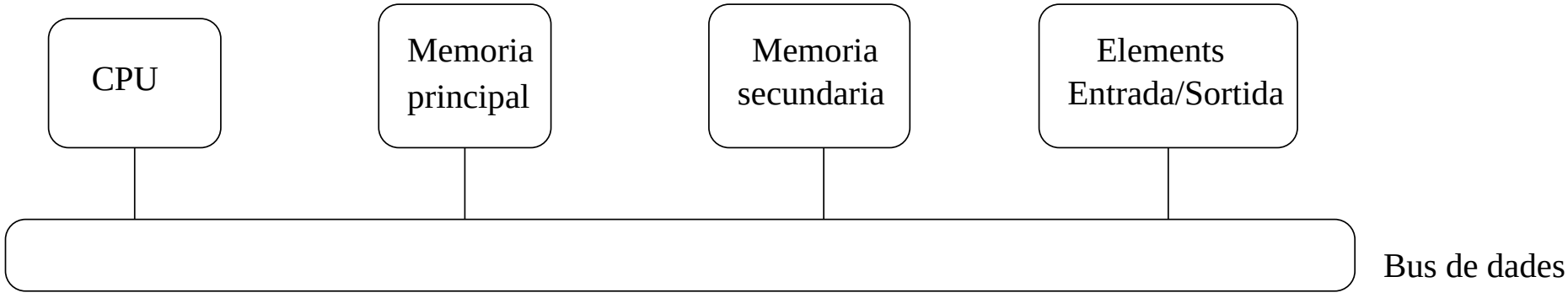
Fitxers

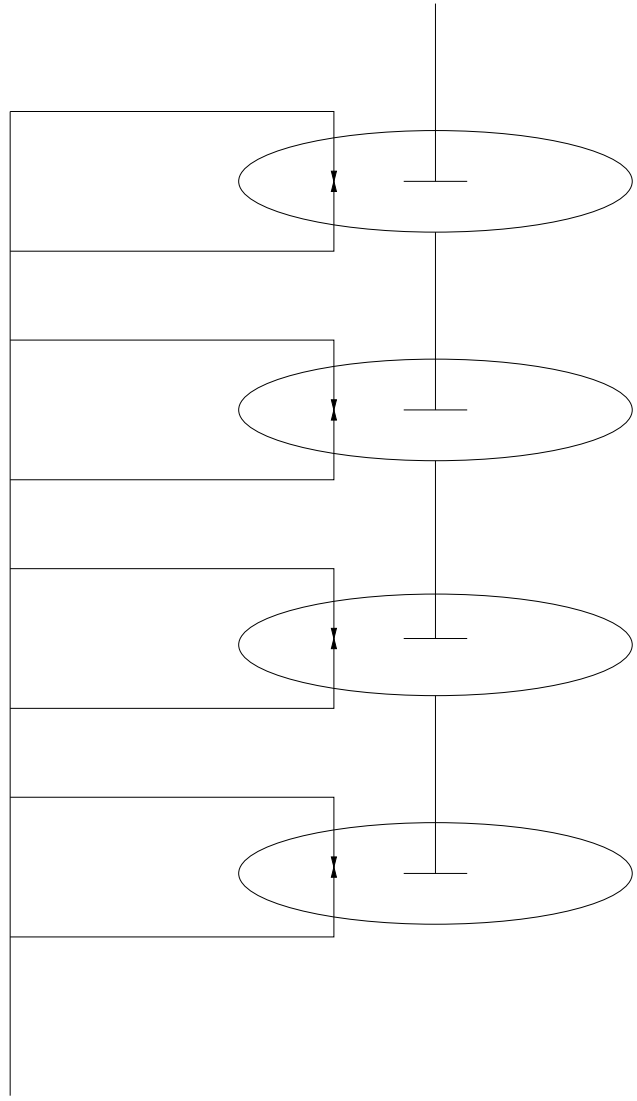
Josep Vilaplana

ETSEIB/GIE

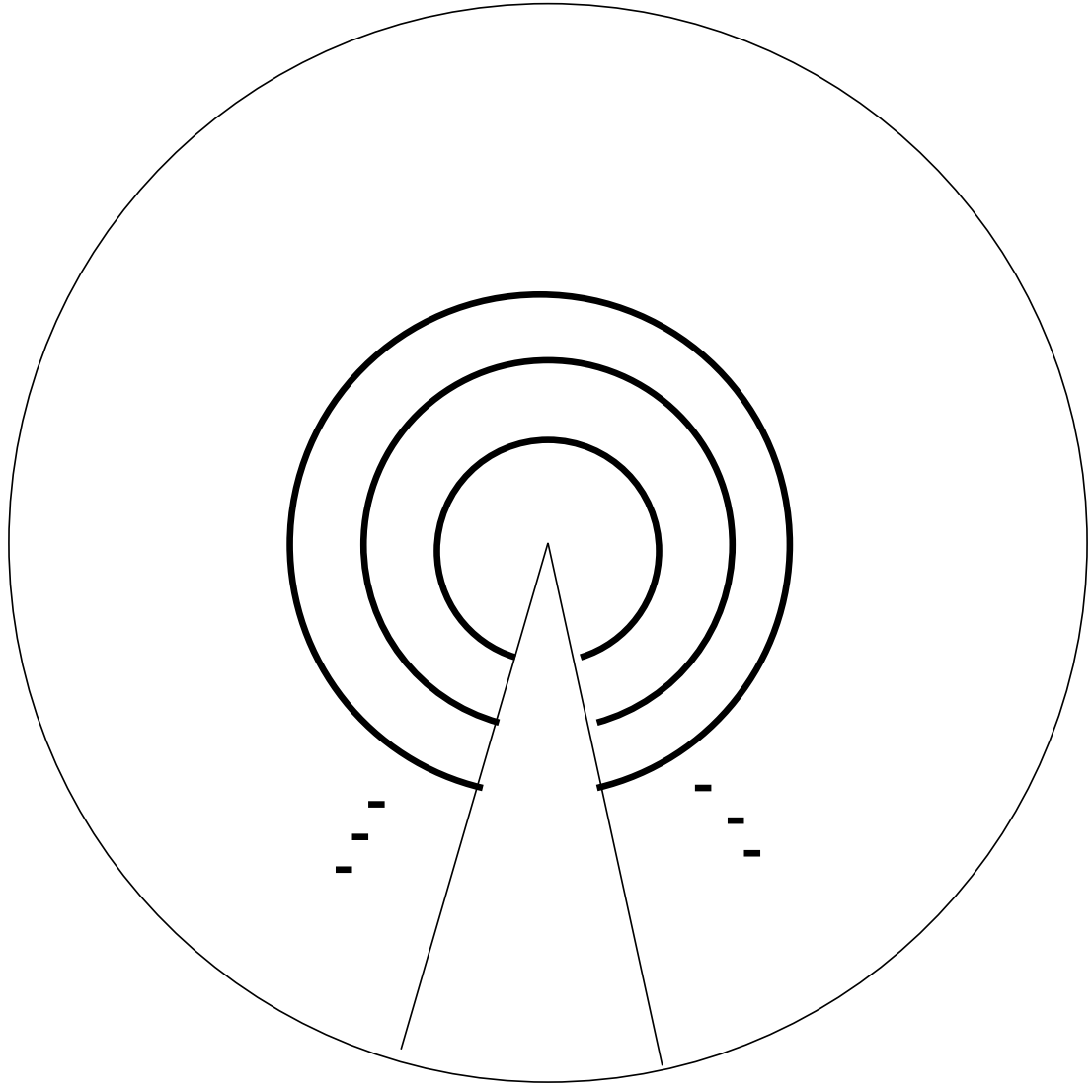
Interconnexió del computador



Esquema d'un disc dur



Esquema d'un disc dur



Especificació Fitxers: tipus *FST*

var *f* : *FST* **fvar**

- FST no és el fitxer. És un tipus que es refereix al fitxer, que és un objecte extern al computador, i guarda l'estat de l'accés a les dades del fitxer.

Especificació Fitxers: tipus *FST*

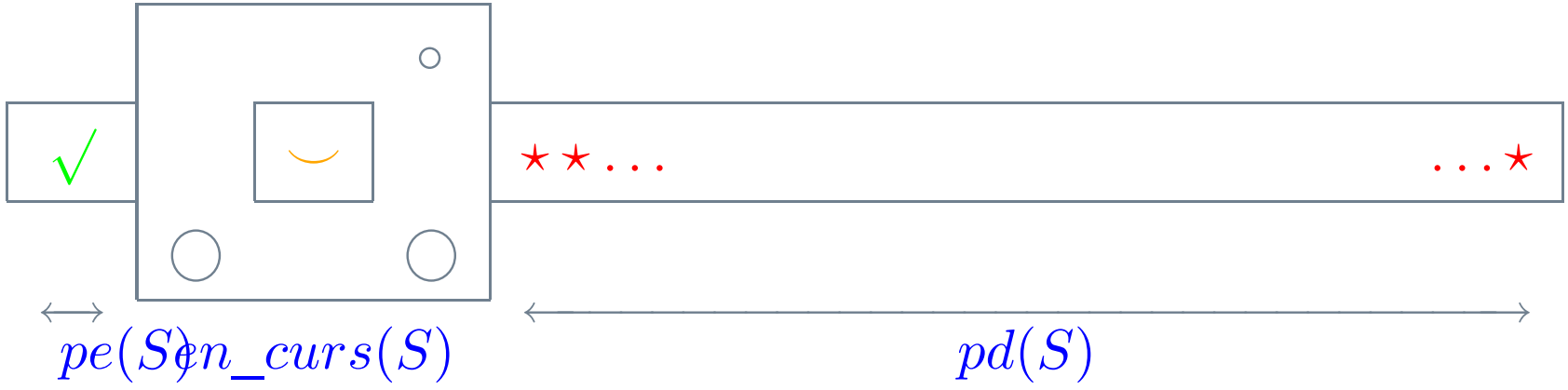
var *f* : *FST* **fvar**

- FST no és el fitxer. És un tipus que es refereix al fitxer, que és un objecte extern al computador, i guarda l'estat de l'accés a les dades del fitxer.
- f és, en l'exemple, el nom de la variable de tipus FST. El nom en sí no té res a veure amb el nom del possible fitxer que es faci servir de f per poder ser accedit.

Accés seqüencial: Inici

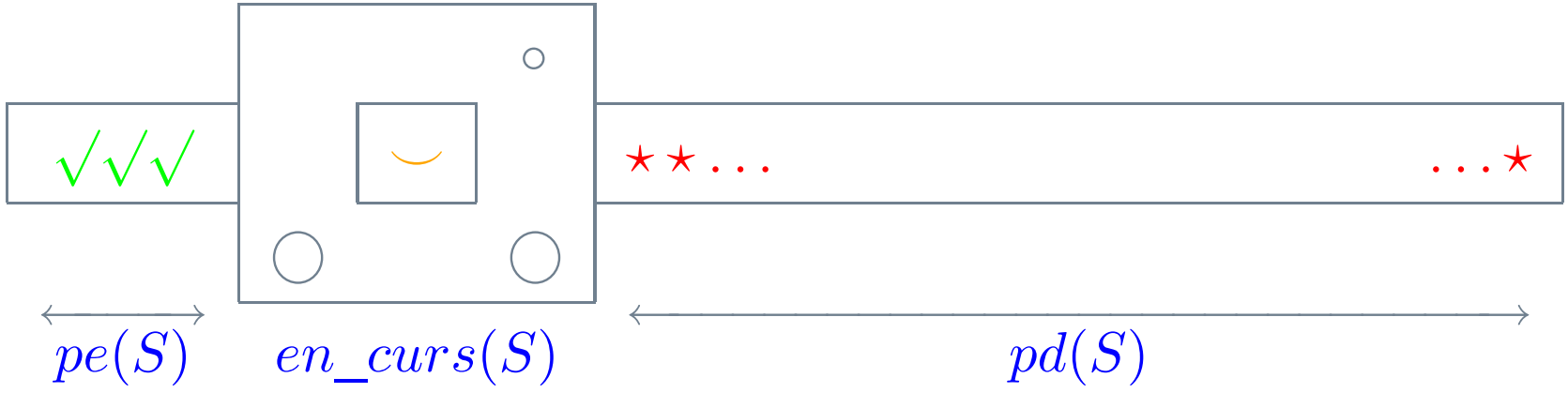
Accés seqüencial: Primer

Següent

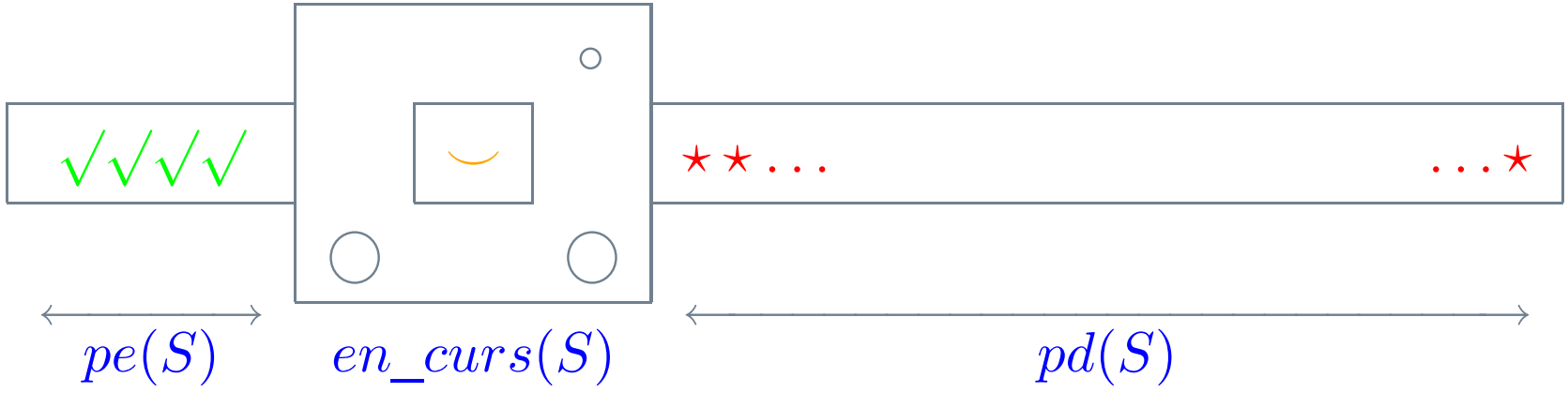


Següent

Següent



Següent



Fi seqüència

Estats al llegir d'una seqüència

- A l'inici, $pe(S) = \emptyset$, $en\_curs(S) = \emptyset$, $S = pd(S)$

Estats al llegir d'una seqüència

- A l'inici, $pe(S) = \emptyset$, $en\_curs(S) = \emptyset$, $S = pd(S)$
- Per a obtenir el següent, *avancem*, i *consultem*

Estats al llegir d'una seqüència

- A l'inici, $pe(S) = \emptyset$, $en\_curs(S) = \emptyset$, $S = pd(S)$
- Per a obtenir el següent, *avancem*, i *consultem*
- Sempre tenim que
  $S = pe(S) \cdot en\_curs(S) \cdot pd(S)$

Estats al llegir d'una seqüència

- A l'inici, $pe(S) = \emptyset$, $en\_curs(S) = \emptyset$, $S = pd(S)$
- Per a obtenir el següent, *avancem*, i *consultem*
- Sempre tenim que
  $S = pe(S) \cdot en\_curs(S) \cdot pd(S)$
- Al final, $S = pe(S)$, $pd(S) = \emptyset$, $en\_curs(S) = \emptyset$

FST: obrir

$\{\mathbf{Pre} : m = \mathit{lectura} \Rightarrow \mathit{existeix}(\mathit{nom})\}$

funció *ObrirFST*$(\mathbf{ent}\ m : \mathit{Mode}, \mathbf{ent}\ \mathit{nom} : \mathit{Cadena})$ **retorna** *FST*

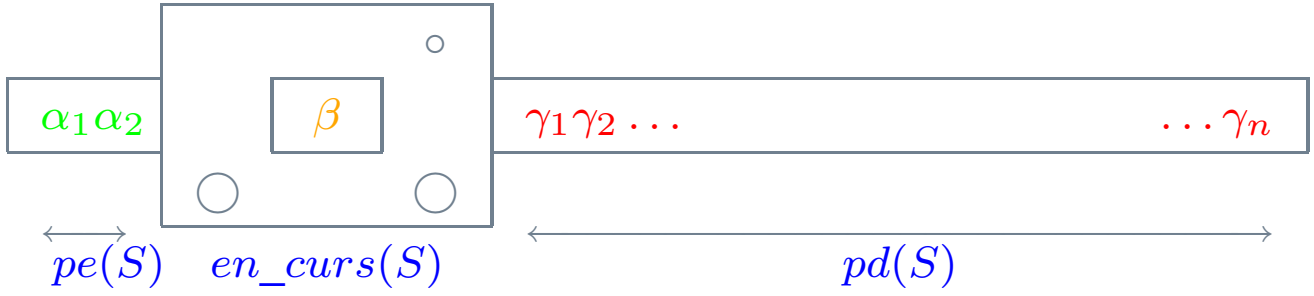
$\{\mathbf{Post} : \mathit{ObrirFST}(m, \mathit{nom}) = f \ \wedge$

$(m = \mathit{lectura} \Rightarrow$

$\quad \mathit{pe}(f) = \mathit{en\_curs}(f) = \emptyset \ \wedge \ \mathit{pd}(f) = \mathit{seqüència}(\mathit{nom})) \ \wedge$

$(m = \mathit{escriptura} \Rightarrow \mathit{pe}(f) = \mathit{en\_curs}(f) = \mathit{pd}(f) = \emptyset)\}$

FST: llegir caràcter



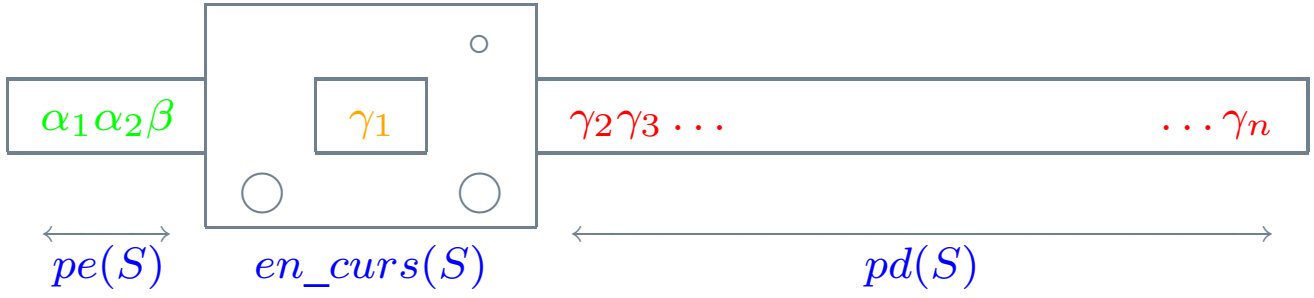
$\{\mathbf{Pre} : \mathit{mode}(f) = \mathit{lectura} \ \wedge$

$\mathit{pe}(f) = \alpha \ \wedge \ \mathit{en\_curs}(f) = \beta \ \wedge \ \mathit{pd}(f) = \gamma \neq \emptyset\}$

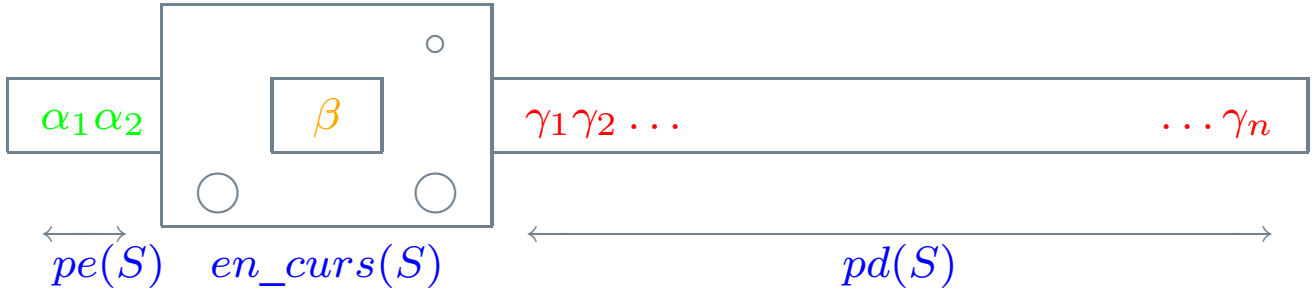
acció *LlegirCaracterFST* (**entsor** f : *FST* , **sor** c : **caràcter**)

$\{\mathbf{Post} : \mathit{mode}(f) = \mathit{lectura} \ \wedge$

$\mathit{pe}(f) = \alpha \cdot \beta \ \wedge \ c = \mathit{en\_curs}(f) = \mathit{cap}(\gamma) \ \wedge \ \mathit{pd}(f) = \mathit{cua}(\gamma)\}$



FST: llegir enter



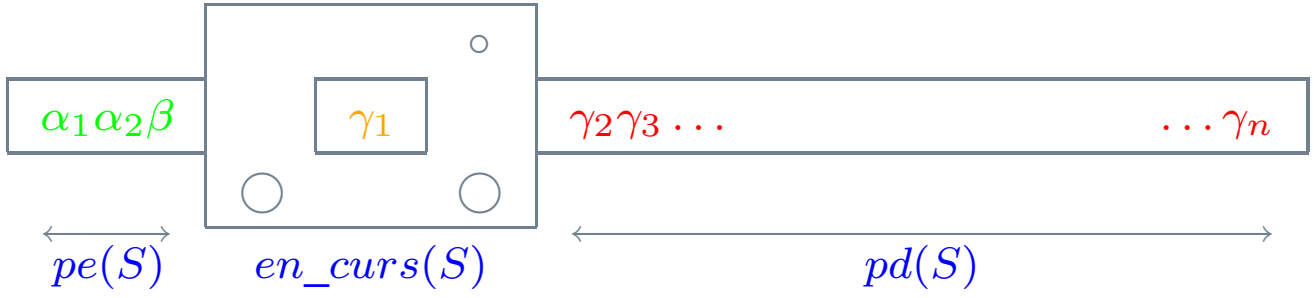
$\{\mathbf{Pre}: mode(f) = lectura \ \wedge$

$pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \neq \emptyset\}$

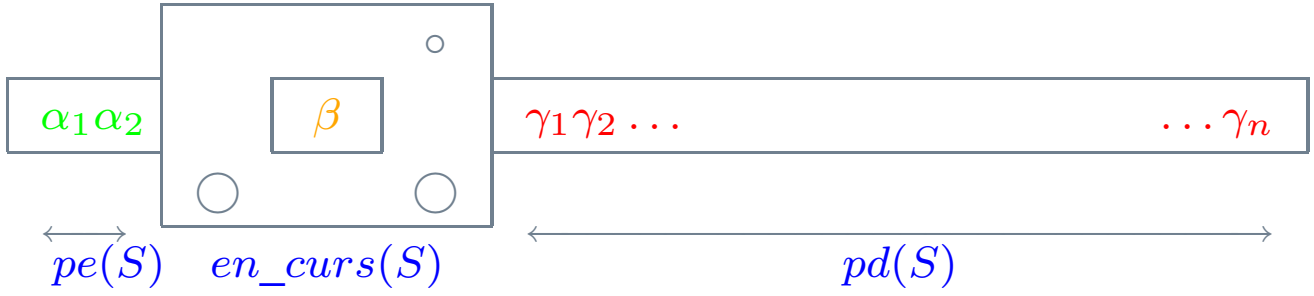
acció *LlegirEnterFST* (**entsor** f : *FST* , **sor** e : **enter**)

$\{\mathbf{Post}: mode(f) = lectura \ \wedge$

$pe(f) = \alpha \cdot \beta \ \wedge \ e = en\_curs(f) = cap(\gamma) \ \wedge \ pd(f) = cua(\gamma)\}$



FST: llegir real



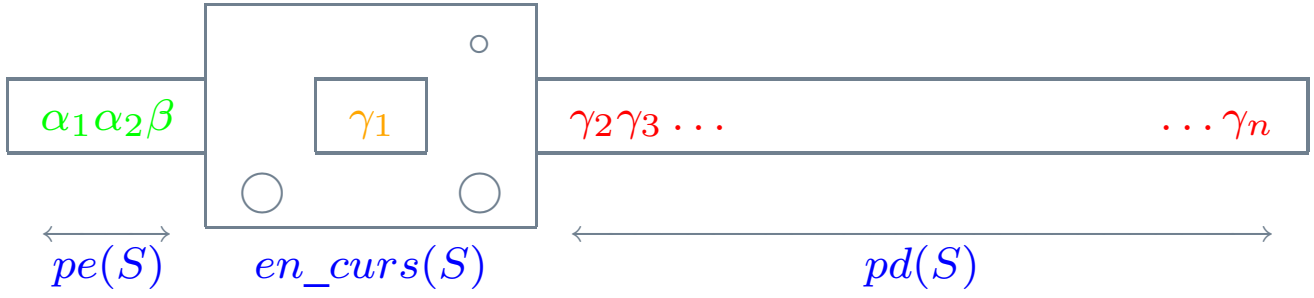
$\{\mathbf{Pre}: mode(f) = lectura \ \wedge$

$pe(f) = \alpha \ \wedge \ en\_curs(f) = \beta \ \wedge \ pd(f) = \gamma \neq \emptyset\}$

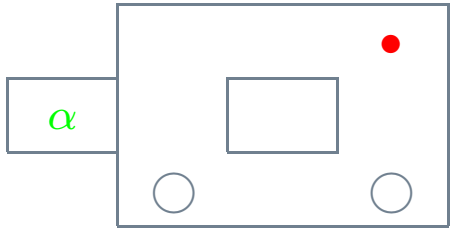
acció *LlegirRealFST* (**entsor** f : *FST*, **sor** r : **real**)

$\{\mathbf{Post}: mode(f) = lectura \ \wedge$

$pe(f) = \alpha \cdot \beta \ \wedge \ r = en\_curs(f) = cap(\gamma) \ \wedge \ pd(f) = cua(\gamma)\}$



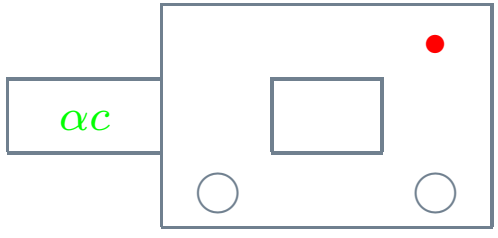
FST: escriure caràcter



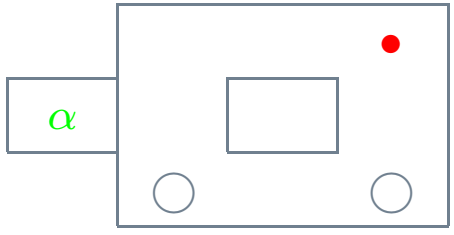
$\{\mathbf{Pre} : \mathit{mode}(f) = \mathit{escriptura} \ \wedge \ \mathit{pe}(f) = \alpha\}$

acció *EscriureCaracterFST*$($**entsor** f : *FST*, **ent** c : **caràcter**$)$

$\{\mathbf{Post} : \mathit{mode}(f) = \mathit{escriptura} \ \wedge \ \mathit{pe}(f) = \alpha \cdot c\}$



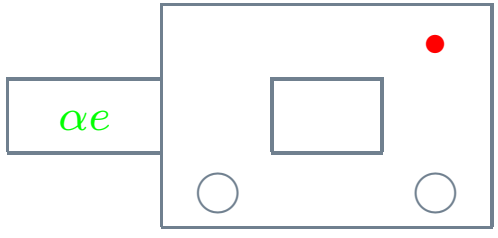
FST: escriure enter



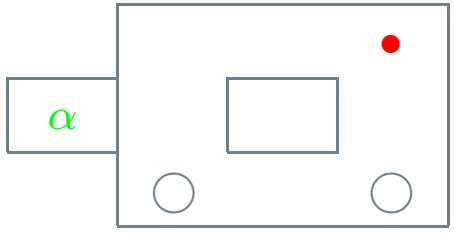
$\{\mathbf{Pre} : \mathit{mode}(f) = \mathit{escriptura} \ \wedge \ \mathit{pe}(f) = \alpha\}$

acció *EscriureEnterFST* $($**entsor** f : *FST*, **ent** e : **enter**$)$

$\{\mathbf{Post} : \mathit{mode}(f) = \mathit{escriptura} \ \wedge \ \mathit{pe}(f) = \alpha \cdot e\}$



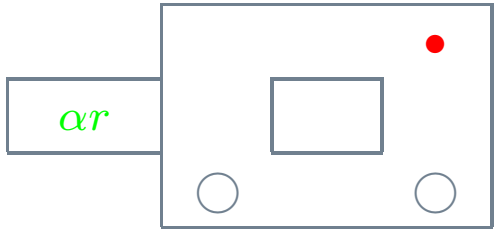
FST: escriure real



$\{\mathbf{Pre} : \mathit{mode}(f) = \mathit{escriptura} \ \wedge \ \mathit{pe}(f) = \alpha\}$

acció *EscriureRealFST*$($**entsor** f : *FST*, **ent** r : **real**$)$

$\{\mathbf{Post} : \mathit{mode}(f) = \mathit{escriptura} \ \wedge \ \mathit{pe}(f) = \alpha \cdot r\}$



FST: fi de fitxer

$\{\mathrm{Pre} : \mathit{mode}(f) = \mathit{lectura}\}$

funció *FdFST*$($**ent** f : *FST*$)$ **retorna booleà**

$\{\mathrm{Post} : \mathit{FdFST}(f) \equiv \mathit{sentinella}(\mathit{en\_curs}(f))\}$

FST: tancar fitxer

$\{\mathbf{Pre} : \mathit{obert}(f)\}$
acció *TancarFST*(**entsor** f : *FST*)
$\{\mathbf{Post} : \neg \mathit{obert}(f)\}$

Exercici: CSE a FST

Dissenyeu un algorisme que guardi en un fitxer FST de nom `"entrada"` una seqüència de caràcters acabada amb el caràcter '.' que hi ha al canal estàndard d'entrada.

Solució: CSE a FST

algorisme $CSEaFST$

var

$\ldots\ldots$

fvar

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**...

fvar

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**...

fvar

$iniciTractament$

$LlegirCaracter(c)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**...

fvar

$iniciTractament$

$LlegirCaracter(c)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$LlegirCaracter(c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**...

fvar

$iniciTractament$

$LlegirCaracter(c)$

mentre $c \neq$ '**.**' **fer**

$tractarElement$

$LlegirCaracter(c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**...

fvar

$iniciTractament$

$LlegirCaracter(c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracterFST(f, c)$

$LlegirCaracter(c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**; $f : FST$

fvar

$iniciTractament$

$LlegirCaracter(c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracterFST(f, c)$

$LlegirCaracter(c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**; $f : FST$

fvar

$f := ObrirFST(escriptura, \texttt{"entrada"})$

$LlegirCaracter(c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracterFST(f, c)$

$LlegirCaracter(c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: CSE a FST

algorisme $CSEaFST$

var

c : **caràcter**; $f : FST$

fvar

$f := ObrirFST(escriptura, \texttt{"entrada"})$

$LlegirCaracter(c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracterFST(f, c)$

$LlegirCaracter(c)$

fmentre;

$EscriureCaracterFST(f, c)$

$TancarFST(f)$

falgorisme

Exercici: FST a CSS

Dissenyeu un algorisme que escrigui al canal estàndard de sortida el contingut d'un fitxer FST de nom `"text"` que conté una seqüència de caràcters acabada amb el caràcter '.'.

Solució: FST a CSS

algorisme $FSTaCSS$

var

$\ldots\ldots$

fvar

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

var

c : **caràcter**; f : FST

fvar

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

  var

    c : **caràcter**; $f : FST$

  fvar

  $iniciTractament$

  $f := ObrirFST(lectura, \texttt{"text"})$

  $LlegirCaracterFST(f, c)$

  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $obtenirSeguentElement$

  fmentre;

  $tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

  var

    c : **caràcter**; $f : FST$

  fvar

  $iniciTractament$

  $f := ObrirFST(lectura, \texttt{"text"})$

  $LlegirCaracterFST(f, c)$

  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $LlegirCaracterFST(f, c)$

  fmentre;

  $tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

var

c : **caràcter**; $f : FST$

fvar

$iniciTractament$

$f := ObrirFST(lectura, \texttt{"text"})$

$LlegirCaracterFST(f, c)$

mentre $c \neq$ '**.**' **fer**

$tractarElement$

$LlegirCaracterFST(f, c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

var

c : **caràcter**; $f : FST$

fvar

$iniciTractament$

$f := ObrirFST(lectura, \texttt{"text"})$

$LlegirCaracterFST(f, c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracter(c)$

$LlegirCaracterFST(f, c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

var

c : **caràcter**; $f : FST$

fvar

$iniciTractament$

$f := ObrirFST(lectura, \texttt{"text"})$

$LlegirCaracterFST(f, c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracter(c)$

$LlegirCaracterFST(f, c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

var

c : **caràcter**; $f : FST$

fvar

$f := ObrirFST(lectura, \texttt{"text"})$

$LlegirCaracterFST(f, c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracter(c)$

$LlegirCaracterFST(f, c)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a CSS

algorisme $FSTaCSS$

var

c : **caràcter**; $f : FST$

fvar

$f := ObrirFST(lectura, \texttt{"text"})$

$LlegirCaracterFST(f, c)$

mentre $c \neq$ '**.**' **fer**

$EscriureCaracter(c)$

$LlegirCaracterFST(f, c)$

fmentre;

$TancarFST(f)$

falgorisme

Exercici: FST a FST

Dissenyeu un algorisme que escrigui a un fitxer FST de nom `"desti"` el contingut d'un fitxer FST de nom `"origen"` que conté una seqüència d'enters acabada amb el 0.

Solució: FST a FST

algorisme $FSTaFST$

var

$\ldots\ldots$

fvar

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a FST

algorisme $FSTaFST$

var

e : **enter**; f : $FST \ldots$

fvar

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a FST

algorisme $FSTaFST$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f := ObrirFST(lectura, \texttt{"origen"})$

$LlegirEnterFST(f, e)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a FST

algorisme $FSTaFST$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f := ObrirFST(lectura, \texttt{"origen"})$

$LlegirEnterFST(f, e)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a FST

algorisme $FSTaFST$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f := ObrirFST(lectura, \texttt{"origen"})$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$tractarElement$

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a FST

```
algorisme FSTaFST
  var
    e : enter; f : FST
  fvar
  iniciTractament
  f := ObrirFST(lectura, "origen")
  LlegirEnterFST(f, e)
  mentre e ≠ 0 fer
    EscriureEnterFST(g, e); EscriureCaracterFST(g, ' ');
    LlegirEnterFST(f, e)
  fmentre;
  tractamentFinal

falgorisme
```

Solució: FST a FST

algorisme $FSTaFST$

var

e : **enter**; $f : FST$; $g : FST$

fvar

$iniciTractament$

$f := ObrirFST(lectura, \texttt{"origen"})$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$EscriureEnterFST(g, e)$; $EscriureCaracterFST(g, \texttt{' '})$;

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a FST

algorisme $FSTaFST$

var

e : **enter**; $f : FST$; $g : FST$

fvar

$g := ObrirFST(escriptura, \texttt{"desti"})$

$f := ObrirFST(lectura, \texttt{"origen"})$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$EscriureEnterFST(g, e)$; $EscriureCaracterFST(g, \texttt{' '})$;

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

falgorisme

Solució: FST a FST

algorisme $FSTaFST$

var

e : **enter**; $f : FST$; $g : FST$

fvar

$g := ObrirFST(escriptura, \texttt{"desti"})$

$f := ObrirFST(lectura, \texttt{"origen"})$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$EscriureEnterFST(g, e)$; $EscriureCaracterFST(g, \texttt{' '})$;

$LlegirEnterFST(f, e)$

fmentre;

$EscriureEnterFST(g, e)$

$TancarFST(f)$

$TancarFST(g)$

falgorisme

Exercici: Acció de còpia de FST a FS

Dissenyeu una acció, que donats dos noms de fitxers, copii el contingut del fitxer identificat amb el primer nom, al fitxer que té el segon nom.
El contingut del fitxer FST a copiar conté una seqüència d'enters acabada amb el 0.

Solució: Acció de copiar FST's

acció $copia(...)$

$.....$

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

facció

Solució: Acció de copiar FST's

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

$\ldots\ldots$

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

facció

Solució: Acció de copiar FST's

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    .....
  fvar
  iniciTractament

  obtenirPrimerElement
  mentre ¬darrerElement fer
    tractarElement
    obtenirSeguentElement
  fmentre;
  tractamentFinal


facció
```

Solució: Acció de copiar FST's

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    e : enter; f : FST...
  fvar
  iniciTractament

  obtenirPrimerElement
  mentre ¬darrerElement fer
    tractarElement
    obtenirSeguentElement
  fmentre;
  tractamentFinal


facció
```

Solució: Acció de copiar FST's

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

facció

Solució: Acció de copiar FST's

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

facció

Solució: Acció de copiar FST's

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$tractarElement$

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

facció

Solució: Acció de copiar FST's

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    e : enter; f : FST
  fvar
  iniciTractament
  f = ObrirFST(lectura, f1)
  LlegirEnterFST(f, e)
  mentre e ≠ 0 fer
    EscriureEnterFST(g, e); EscriureCaracterFST(g, ' ');
    LlegirEnterFST(f, e)
  fmentre;
  tractamentFinal

facció
```

Solució: Acció de copiar FST's

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    e : enter; f : FST; g : FST
  fvar
  iniciTractament
  f = ObrirFST(lectura, f1)
  LlegirEnterFST(f, e)
  mentre e ≠ 0 fer
    EscriureEnterFST(g, e); EscriureCaracterFST(g, ' ');
    LlegirEnterFST(f, e)
  fmentre;
  tractamentFinal

facció
```

Solució: Acció de copiar FST's

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST$; $g : FST$

fvar

$g = ObrirFST(escriptura, f2)$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$EscriureEnterFST(g, e)$; $EscriureCaracterFST(g, ' ')$;

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

facció

Solució: Acció de copiar FST's

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST$; $g : FST$

fvar

$g = ObrirFST(escriptura, f2)$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$EscriureEnterFST(g, e)$; $EscriureCaracterFST(g, ' ')$;

$LlegirEnterFST(f, e)$

fmentre;

$EscriureEnterFST(g, e)$

$TancarFST(f)$

$TancarFST(g)$

facció

És també una solució?

acció $copia(...)$

$\ldots\ldots$

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

facció

És també una solució?

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

. . .. . .

fvar

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

facció

És també una solució?

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    e : enter; f : FST...
  fvar
  iniciTractament

  obtenirPrimerElement
  mentre ¬darrerElement fer
    tractarElement
    obtenirSeguentElement
  fmentre;
  tractamentFinal


facció
```

És també una solució?

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre;

$tractamentFinal$

facció

És també una solució?

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

facció

És també una solució?

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST \ldots$

fvar

$iniciTractament$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$tractarElement$

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

facció

És també una solució?

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    e : enter; f : FST
  fvar
  iniciTractament
  f = ObrirFST(lectura, f1)
  LlegirEnterFST(f, e)
  mentre e ≠ 0 fer
    EscriureEnterFST(g, e); EscriureCaracterFST(g, ' ');
    LlegirEnterFST(f, e)
  fmentre;
  tractamentFinal

facció
```

És també una solució?

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    e : enter; f : FST; g : FST
  fvar
  iniciTractament
  f = ObrirFST(lectura, f1)
  LlegirEnterFST(f, e)
  mentre e ≠ 0 fer
    EscriureEnterFST(g, e); EscriureCaracterFST(g, ' ');
    LlegirEnterFST(f, e)
  fmentre;
  tractamentFinal

facció
```

És també una solució?

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

var

e : **enter**; $f : FST$; $g : FST$

fvar

$g = ObrirFST(escriptura, f2)$

$f = ObrirFST(lectura, f1)$

$LlegirEnterFST(f, e)$

mentre $e \neq 0$ **fer**

$EscriureEnterFST(g, e)$; $EscriureCaracterFST(g, ' ')$;

$LlegirEnterFST(f, e)$

fmentre;

$tractamentFinal$

facció

És també una solució?

```
acció copia(ent f1 : Cadena, ent f2 : Cadena)
  var
    e : enter; f : FST; g : FST
  fvar
  g = ObrirFST(escriptura, f2)
  f = ObrirFST(lectura, f1)
  LlegirEnterFST(f, e)
  mentre e ≠ 0 fer
    EscriureEnterFST(g, e); EscriureCaracterFST(g, ' ');
    LlegirEnterFST(f, e)
  fmentre;
  EscriureEnterFST(g, e)
  TancarFST(f)
  TancarFST(g)
facció
```

Exemple de crida

L'anterior exercici es resumiria en aquest algorisme:

algorisme $FSTaFST$

$copia($`"origen"`$,$`"desti"`$)$

falgorisme

Exercici: Acció de còpia de FST a FS

Dissenyeu un algorisme que donats dos noms de fitxers introduïts pel canal estàndard d'entrada, copii el contingut del fitxer identificat amb el primer nom, al fitxer que té el segon nom. El contingut del fitxer FST a copiar conté una seqüència d'enters acabada amb el 0.

Es disposa del següent:

acció $copia($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena)$

tipus $tNom$ **ftipus**

acció $llegirNom($**sor** $: nom : tNom)$

$\{$**Pre:** A l'entrada es té una seqüència de caracters s que es pot veure com una seqüència de noms n, on cada nom és una subseqüència de caràcters diferents a l'espai acabada amb un espai. I tenim $pe(n) = PE$, $en\_curs(n) = L$,$(pd(s) = N \cdot PD)$, i $pd(n) \neq \emptyset$, i N i L són subseqüències de caràcters acabada amb el caràcter espai que representen noms$\}$

$\{$**Post:** nom representa N, $pe(n) = PE \cdot L$, $en\_curs(n) = N$ i $pd(n) = PD\}$

funció $NomACadena($**ent** $n : tNom)$ **retorna** $Cadena$

$\{$**Pre:** n representa N com a nom$\}$

$\{$**Post:** $NomACadena(n)$ representa N com a $Cadena\}$

Solució: Acció de copiar FST's

algorisme $FSTaFST$

var

$nom1, nom2 : tNom$

fvar

$llegirNom(nom1)$

$llegirNom(nom2)$

$copia(NomACadena(nom1), NomACadena(nom2))$

falgorisme

Intersecció de fitxers

Dissenyar una acció que donats tres noms de fitxers FST $n1$, $n2$, i $n3$, escrigui en el fitxer de nom $n3$ la intersecció dels fitxers de nom $n1$ i $n2$ respectivament. Els fitxers $n1$ i $n2$ contenen una seqüència de $tRegistre$ acabada amb un registre sentinella.

Nota: En un fitxer no hi han registres repetits.

Podeu comptar amb que ja estan definits i desenvolupats:

tipus $tRegistre$ **ftipus**

acció $llegirRegistreFST($**entsor** $f : FST,$ **sor** $r : tRegistre)$

acció $escriureRegistreFST($**entsor** $f : FST,$ **ent** $r : tRegistre)$

funció $registreSentinella($**ent** $r : tRegistre)$ **retorna booleà**

funció $registresIguals($**ent** $r1 : tRegistre,$ **ent** $r2 : tRegistre)$ **retorna booleà**

Solució intersecció

acció $interseca(...)$
  var . . .. . . **fvar**
  $iniciTractament$
  $obtenirPrimerElement$

  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $obtenirSeguentElement$
  fmentre
  $tractamentFinal$
facció

Solució intersecció

acció $interseca($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena,$ **ent** $f3 : Cadena)$

var . . .. . . **fvar**

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre

$tractamentFinal$

facció

Solució intersecció

acció $interseca($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena,$ **ent** $f3 : Cadena)$

var . . .. . . **fvar**

$iniciTractament$

$obtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$obtenirSeguentElement$

fmentre

$tractamentFinal$

facció

Solució intersecció

```
acció interseca(ent f1 : Cadena, ent f2 : Cadena, ent f3 : Cadena)
  var rf : tRegistre;  f : FST... fvar
  iniciTractament
  obtenirPrimerElement

  mentre ¬darrerElement fer



    tractarElement



    obtenirSeguentElement
  fmentre
  tractamentFinal
facció
```

Solució intersecció

acció $interseca($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena,$ **ent** $f3 : Cadena)$

  var $rf : tRegistre;\ f : FST \ldots$ **fvar**

  $iniciTractament$

  $f := ObrirFST(lectura, f1)$

  $llegirRegistreFST(f, rf)$

  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $obtenirSeguentElement$

  fmentre

  $tractamentFinal$

facció

Solució intersecció

acció $interseca($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena,$ **ent** $f3 : Cadena)$

var $rf : tRegistre;\ f : FST...$ **fvar**

$iniciTractament$

$f := ObrirFST(lectura, f1)$

$llegirRegistreFST(f, rf)$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$llegirRegistreFST(f, rf)$

fmentre

$tractamentFinal$

facció

Solució intersecció

acció $interseca($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena,$ **ent** $f3 : Cadena)$

  var $rf : tRegistre;\ f : FST \ldots$ **fvar**

  $iniciTractament$

  $f := ObrirFST(lectura, f1)$

  $llegirRegistreFST(f, rf)$

  mentre $\neg registreSentinella(rf)$ **fer**

    $tractarElement$

    $llegirRegistreFST(f, rf)$

  fmentre

  $tractamentFinal$

facció

Solució intersecció

```
acció interseca(ent f1 : Cadena, ent f2 : Cadena, ent f3 : Cadena)
  var rf : tRegistre;  f : FST fvar
  iniciTractament
  f := ObrirFST(lectura, f1)
  llegirRegistreFST(f, rf)
  mentre ¬registreSentinella(rf) fer
    g := ObrirFST(lectura, f2)
    llegirRegistreFST(g, rg);  trobat := fals;
    mentre ¬registreSentinella(rg) ∧ ¬trobat fer
      si registresIguals(rf, rg) → trobat := cert
      □ ¬registresIguals(rf, rg) → llegirRegistreFST(g, rg)
      fsi
    fmentre
    si trobat → escriureRegistreFST(h, rf) □ ¬trobat → fsi;  TancarFST(g)
    llegirRegistreFST(f, rf)
  fmentre
  tractamentFinal
facció
```

Solució intersecció

```
acció interseca(ent f1 : Cadena, ent f2 : Cadena, ent f3 : Cadena)
  var rf, rg : tRegistre; f, g, h : FST; trobat : booleà fvar
  iniciTractament
  f := ObrirFST(lectura, f1)
  llegirRegistreFST(f, rf)
  mentre ¬registreSentinella(rf) fer
    g := ObrirFST(lectura, f2)
    llegirRegistreFST(g, rg); trobat := fals;
    mentre ¬registreSentinella(rg) ∧ ¬trobat fer
      si registresIguals(rf, rg) → trobat := cert
      □ ¬registresIguals(rf, rg) → llegirRegistreFST(g, rg)
      fsi
    fmentre
    si trobat → escriureRegistreFST(h, rf) □ ¬trobat → fsi; TancarFST(g)
    llegirRegistreFST(f, rf)
  fmentre
  tractamentFinal
facció
```

Solució intersecció

acció $interseca($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena,$ **ent** $f3 : Cadena)$

var $rf, rg : tRegistre;\ f, g, h : FST;\ trobat :$ **booleà fvar**

$h := ObrirFST(escriptura, f3)$

$f := ObrirFST(lectura, f1)$

$llegirRegistreFST(f, rf)$

mentre $\neg registreSentinella(rf)$ **fer**

$g := ObrirFST(lectura, f2)$

$llegirRegistreFST(g, rg);\ trobat :=$ **fals**;

mentre $\neg registreSentinella(rg) \wedge \neg trobat$ **fer**

si $registresIguals(rf, rg) \rightarrow trobat :=$ **cert**

$\square\ \neg registresIguals(rf, rg) \rightarrow llegirRegistreFST(g, rg)$

fsi

fmentre

si $trobat \rightarrow escriureRegistreFST(h, rf)\ \square\ \neg trobat \rightarrow$ **fsi**; $TancarFST(g)$

$llegirRegistreFST(f, rf)$

fmentre

$tractamentFinal$

facció

Solució intersecció

acció $interseca($**ent** $f1 : Cadena,$ **ent** $f2 : Cadena,$ **ent** $f3 : Cadena)$
  var $rf, rg : tRegistre;\ f, g, h : FST;\ trobat :$ **booleà fvar**
  $h := ObrirFST(escriptura, f3)$
  $f := ObrirFST(lectura, f1)$
  $llegirRegistreFST(f, rf)$
  mentre $\neg registreSentinella(rf)$ **fer**
    $g := ObrirFST(lectura, f2)$
    $llegirRegistreFST(g, rg);\ trobat :=$ **fals**;
    mentre $\neg registreSentinella(rg) \wedge \neg trobat$ **fer**
      si $registresIguals(rf, rg) \rightarrow trobat :=$ **cert**
      $\square\ \neg registresIguals(rf, rg) \rightarrow llegirRegistreFST(g, rg)$
      fsi
    fmentre
    si $trobat \rightarrow escriureRegistreFST(h, rf)\ \square\ \neg trobat \rightarrow$ **fsi**; $TancarFST(g)$
    $llegirRegistreFST(f, rf)$
  fmentre
  $escriureRegistreFST(h, rf);\ TancarFST(f);\ TancarFST(h)$
facció

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1)\ \wedge\ \neg RegSentinella(r2)$ **fer**

si

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1)\ \wedge\ \neg RegSentinella(r2)$ **fer**

si $ComparaRegs(r1, r2) < 0 \rightarrow EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$

□

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1)\ \wedge\ \neg RegSentinella(r2)$ **fer**

si $ComparaRegs(r1, r2) < 0 \rightarrow EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$

$\square\ ComparaRegs(r1, r2) > 0 \rightarrow EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$

$\square$

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1)\ \wedge\ \neg RegSentinella(r2)$ **fer**

si $ComparaRegs(r1, r2) < 0 \rightarrow EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$

$\square\ ComparaRegs(r1, r2) > 0 \rightarrow EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$

$\square\ ComparaRegs(r1, r2) = 0 \rightarrow EscriureRegFST(f3, r1)$

$LlegirRegFST(f1, r1)$

$LlegirRegFST(f2, r2)$

fsi

fmentre

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1) \ \wedge \ \neg RegSentinella(r2)$ **fer**

si $ComparaRegs(r1, r2) < 0 \rightarrow EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$

$\square\ ComparaRegs(r1, r2) > 0 \rightarrow EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$

$\square\ ComparaRegs(r1, r2) = 0 \rightarrow EscriureRegFST(f3, r1)$

$LlegirRegFST(f1, r1)$

$LlegirRegFST(f2, r2)$

fsi

fmentre

mentre $\neg RegSentinella(r1)$ **fer** $EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$ **fmentre**

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1) \ \wedge \ \neg RegSentinella(r2)$ **fer**

si $ComparaRegs(r1, r2) < 0 \rightarrow EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$

$\square\ ComparaRegs(r1, r2) > 0 \rightarrow EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$

$\square\ ComparaRegs(r1, r2) = 0 \rightarrow EscriureRegFST(f3, r1)$

$LlegirRegFST(f1, r1)$

$LlegirRegFST(f2, r2)$

fsi

fmentre

mentre $\neg RegSentinella(r1)$ **fer** $EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$ **fmentre**

mentre $\neg RegSentinella(r2)$ **fer** $EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$ **fmentre**

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1) \ \wedge \ \neg RegSentinella(r2)$ **fer**

si $ComparaRegs(r1, r2) < 0 \rightarrow EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$

$\square\ ComparaRegs(r1, r2) > 0 \rightarrow EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$

$\square\ ComparaRegs(r1, r2) = 0 \rightarrow EscriureRegFST(f3, r1)$

$LlegirRegFST(f1, r1)$

$LlegirRegFST(f2, r2)$

fsi

fmentre

mentre $\neg RegSentinella(r1)$ **fer** $EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$ **fmentre**

mentre $\neg RegSentinella(r2)$ **fer** $EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$ **fmentre**

$EscriureRegFST(f3, r1)$

Algorisme fusió

algorisme $Fusio$

var $f1, f2, f3 : FST;\ r1, r2 : tReg$ **fvar**

$f1 := ObrirFST(lectura, \text{“A”});\ f2 := ObrirFST(lectura, \text{“B”});$

$f3 := ObrirFST(escriptura, \text{“C”})$

$LlegirRegFST(f1, r1);\ LlegirRegFST(f2, r2)$

mentre $\neg RegSentinella(r1)\ \wedge\ \neg RegSentinella(r2)$ **fer**

si $ComparaRegs(r1, r2) < 0 \rightarrow EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$

$\square\ ComparaRegs(r1, r2) > 0 \rightarrow EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$

$\square\ ComparaRegs(r1, r2) = 0 \rightarrow EscriureRegFST(f3, r1)$

$LlegirRegFST(f1, r1)$

$LlegirRegFST(f2, r2)$

fsi

fmentre

mentre $\neg RegSentinella(r1)$ **fer** $EscriureRegFST(f3, r1);\ LlegirRegFST(f1, r1)$ **fmentre**

mentre $\neg RegSentinella(r2)$ **fer** $EscriureRegFST(f3, r2);\ LlegirRegFST(f2, r2)$ **fmentre**

$EscriureRegFST(f3, r1)$

$TancarFST(f1);\ TancarFST(f2);\ TancarFST(f3)$

falgorisme

Intersecció de fitxers

Dissenyar una acció que donats tres noms de fitxers FST $n1$, $n2$, i $n3$, escrigui en el fitxer de nom $n3$ la intersecció dels fitxers de nom $n1$ i $n2$ respectivament. Els fitxers $n1$ i $n2$ contenen una seqüència de $tRegistre$ acabada amb un registre sentinella. **Els registres d'ambdós fitxers estan ordenats.**

Nota: En un fitxer no hi han registres repetits.

Podeu comptar amb que ja estan definits i desenvolupats:

tipus $tRegistre$ **ftipus**

acció $llegirRegistreFST($**entsor** $f : FST,$ **sor** $r : tRegistre)$

acció $escriureRegistreFST($**entsor** $f : FST,$ **ent** $r : tRegistre)$

funció $registreSentinella($**ent** $r : tRegistre)$ **retorna booleà**

funció $registresMenor($**ent** $r1 : tRegistre,$ **ent** $r2 : tRegistre)$ **retorna booleà**

funció $registresIguals($**ent** $r1 : tRegistre,$ **ent** $r2 : tRegistre)$ **retorna booleà**