1.1 Conceptos básicos: Naturaleza no determinística de los algoritmos concurrentes

- Considere la ejecución del siguiente algoritmo concurrente en un monoprocesador

  {P || Q }

  Donde P es x = x + 1 y Q es x = x - 1
- Quizá pensemos que las únicas dos opciones son que se ejecute primero P y después Q o visceversa, es decir que se tenga una de las dos ejecuciones secuenciales siguientes: PQ ó QP y que siempre se tiene el resultado correcto, es decir, el valor de x no cambia.
- Aquí estamos cometiendo el error de considerar que las instrucciones de alto nivel son atómicas (es decir que se ejecutan totalmente una vez que inician.

Elizabeth Pérez Cortés

1.1 Conceptos básicos: Naturaleza no determinística de los algoritmos concurrentes

- En realidad, las instrucciones anteriores, al ser traducidas por el compilador a lenguaje máquina, se convierten en las siguientes secuencias de instrucciones (las etiquetamos cada instrucción para poder referenciarla después):

```
P1: COPIA R1, x      Q1: COPIA R2, x
P2: INC R1           Q2: DEC R2
P3: COPIA x, R1      Q3: COPIA x, R2
```

Elizabeth Pérez Cortés

1.1 Conceptos básicos: Naturaleza no determinística de los algoritmos concurrentes

- Note que ambas instrucciones trabajan sobre la variable x y que ambas la modifican.
- En otras palabras, x es un recurso compartido
- Para poder visualizar todas las posibles ejecuciones de el programa P || Q, vamos a representar el estado del sistema por el estado de la memoria y los registros.

1.1 Conceptos básicos: Naturaleza no determinística de los algoritmos concurrentes

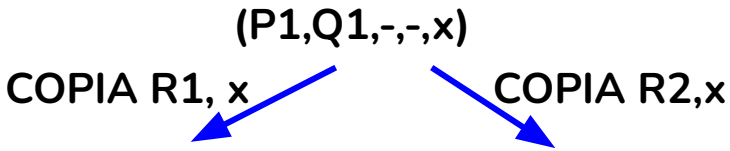
- En nuestro caso el estado de la ejecución concurrente de los dos programas se representa con una tupla:

  (CP1, CP2, R1, R2, x)

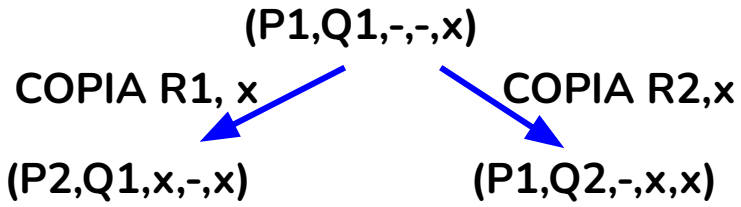
- donde
  - CP1 y CP2 son los contadores de programa
  - R1 y R2 son registros
  - x es la variable compartida.

- El estado inicial es: CP1=P1, CP2=Q1, R1=0, R2=0, x
- ¿En qué orden ocurren las instrucciones cuando ejecutamos P || Q?

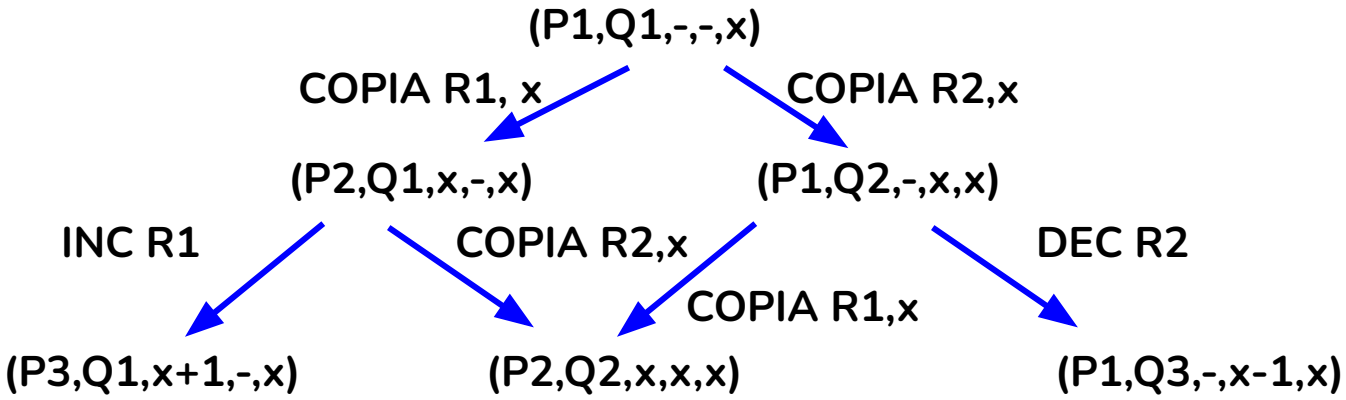
Elizabeth Pérez Cortés

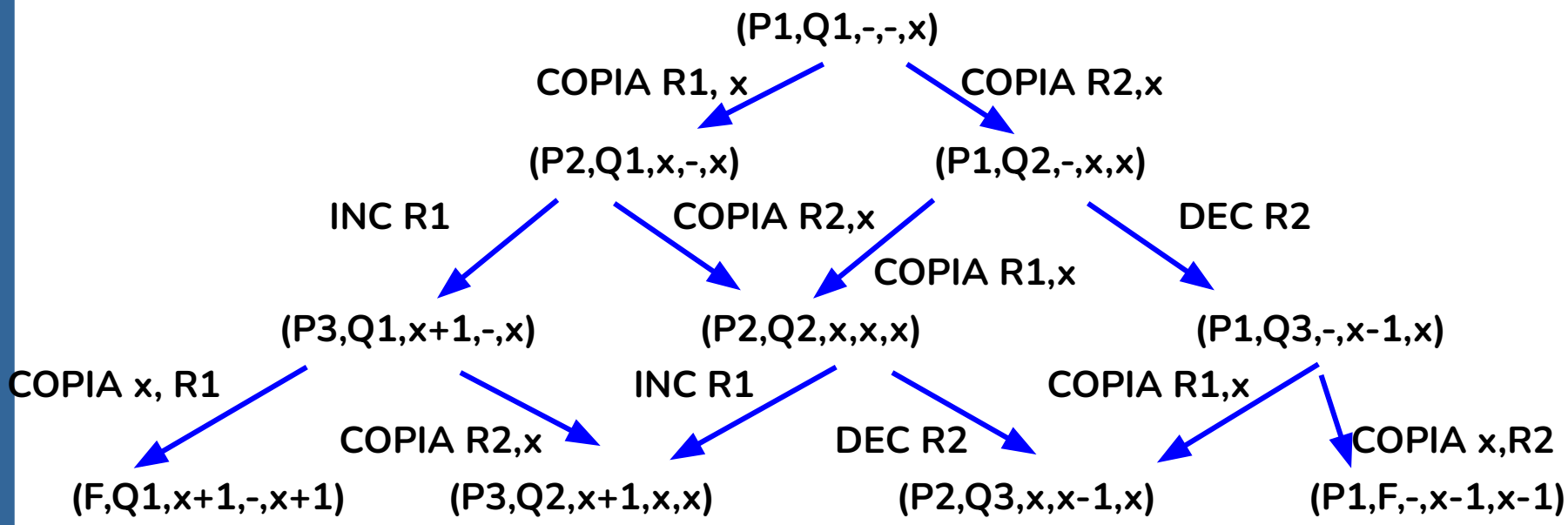


Elizabeth Pérez Cortés

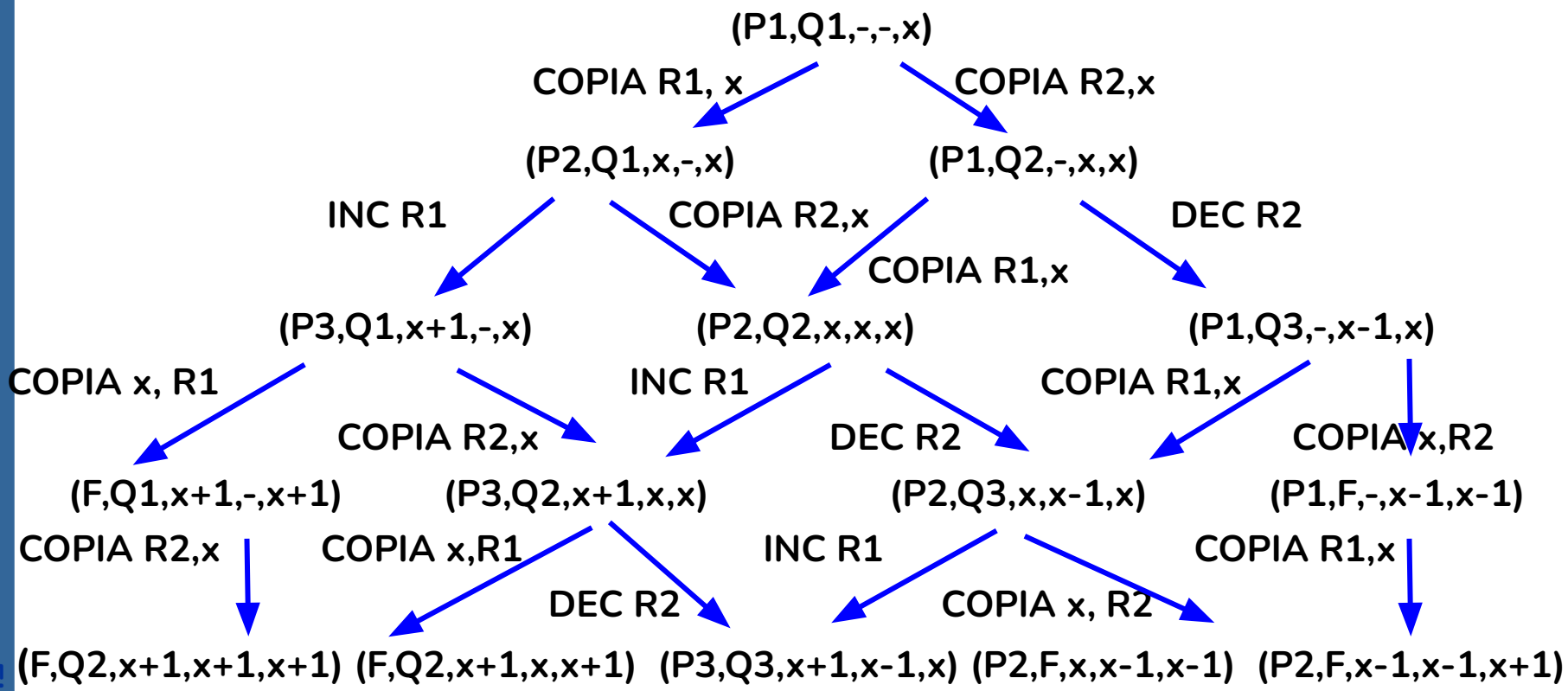


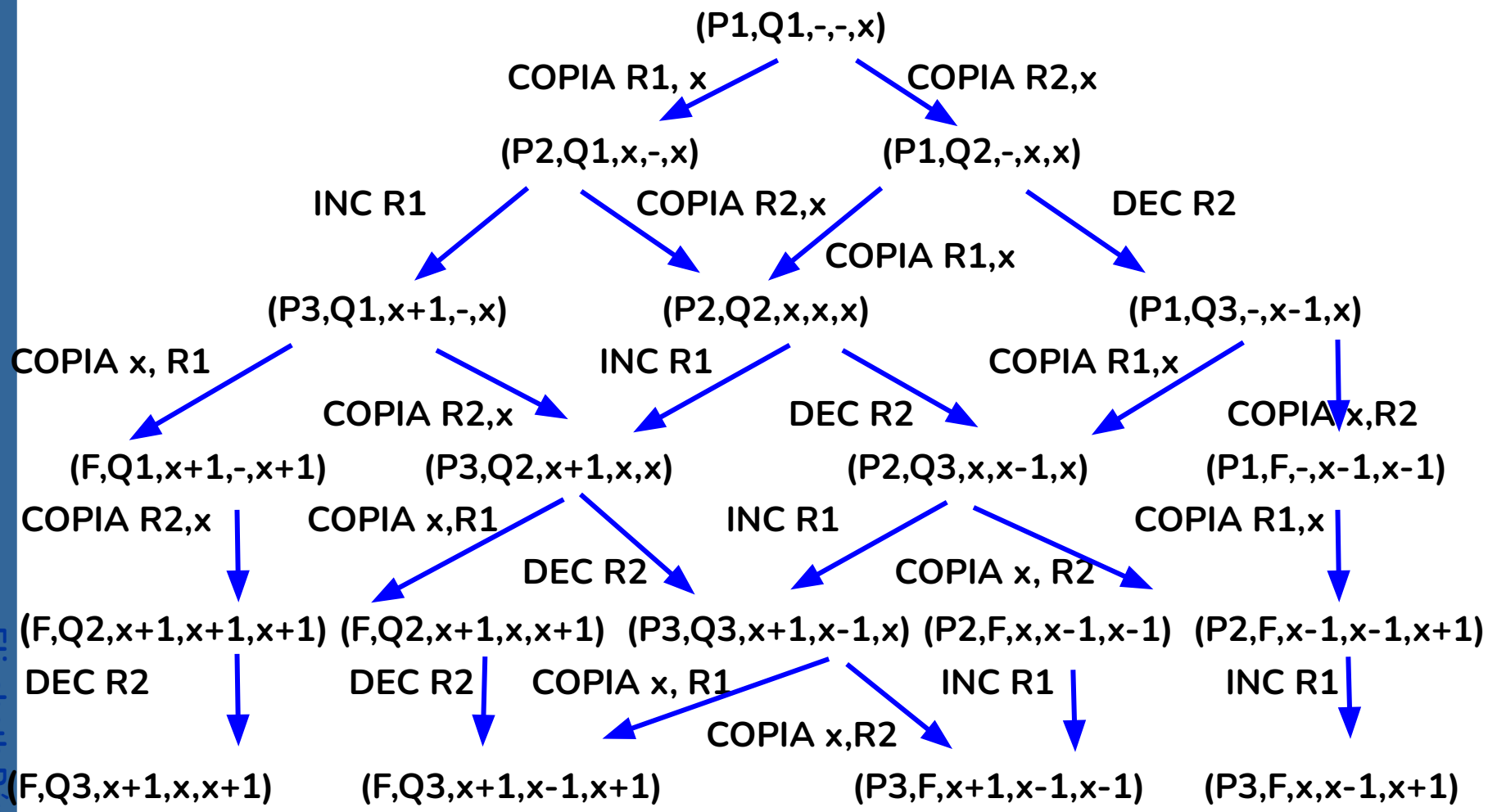
Elizabeth Pérez Cortés


(P1,Q1,-,-,x)
COPIA R1, x
COPIA R2,x
(P2,Q1,x,-,x)
(P1,Q2,-,x,x)
INC R1
COPIA R2,x
COPIA R1,x
DEC R2
(P3,Q1,x+1,-,x)
(P2,Q2,x,x,x)
(P1,Q3,-,x-1,x)

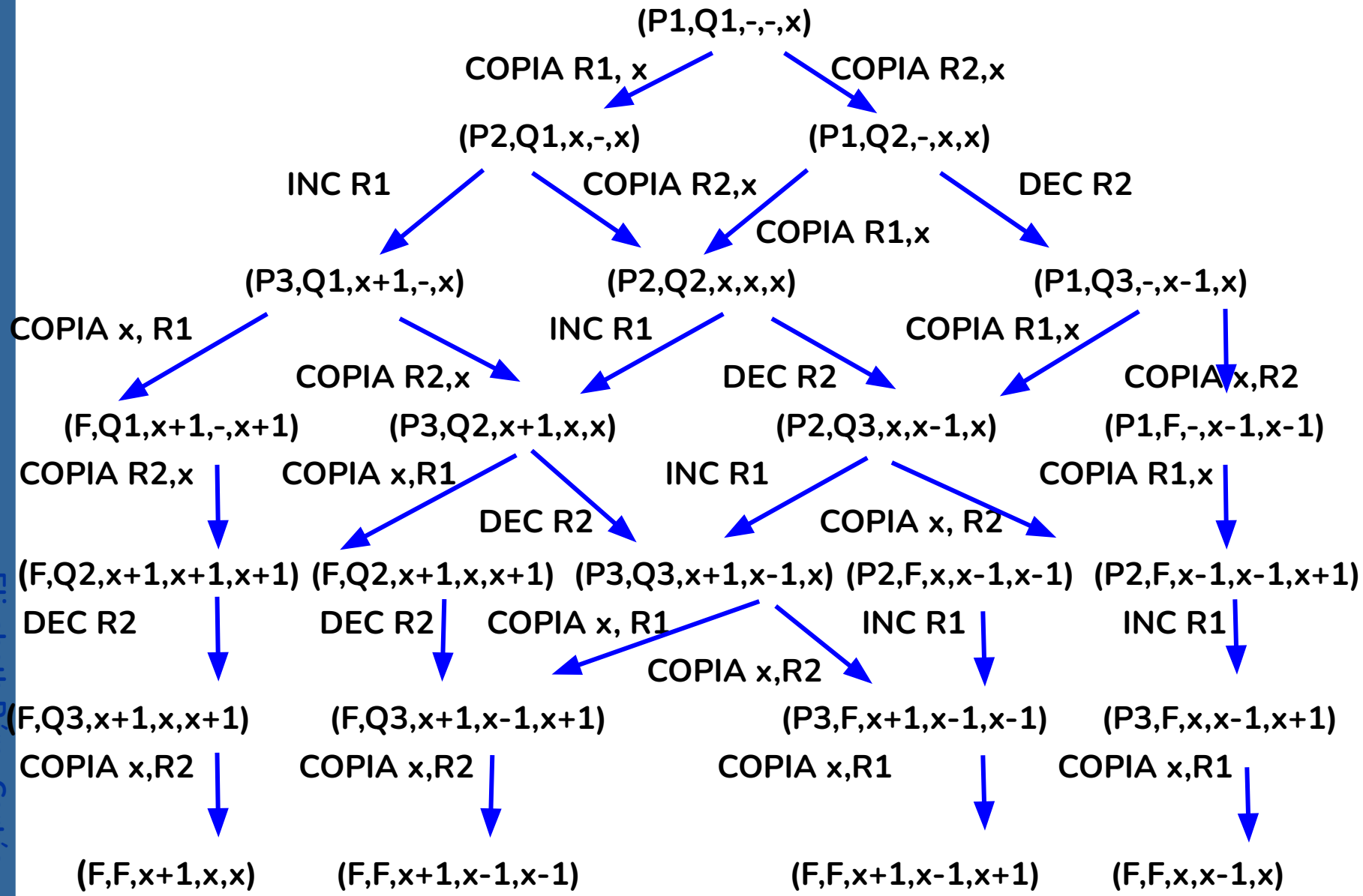


Elizabeth Pérez Cortés


(P1,Q1,-,-,x)
COPIA R1, x
COPIA R2,x
(P2,Q1,x,-,x)
(P1,Q2,-,x,x)
INC R1
COPIA R2,x
DEC R2
COPIA R1,x
(P3,Q1,x+1,-,x)
(P2,Q2,x,x,x)
(P1,Q3,-,x-1,x)
COPIA x, R1
INC R1
COPIA R1,x
COPIA R2,x
DEC R2
COPIA x,R2
(F,Q1,x+1,-,x+1)
(P3,Q2,x+1,x,x)
(P2,Q3,x,x-1,x)
(P1,F,-,x-1,x-1)
COPIA R2,x
COPIA x,R1
INC R1
COPIA R1,x
DEC R2
COPIA x, R2
(F,Q2,x+1,x+1,x+1)
(F,Q2,x+1,x,x+1)
(P3,Q3,x+1,x-1,x)
(P2,F,x,x-1,x-1)
(P2,F,x-1,x-1,x+1)



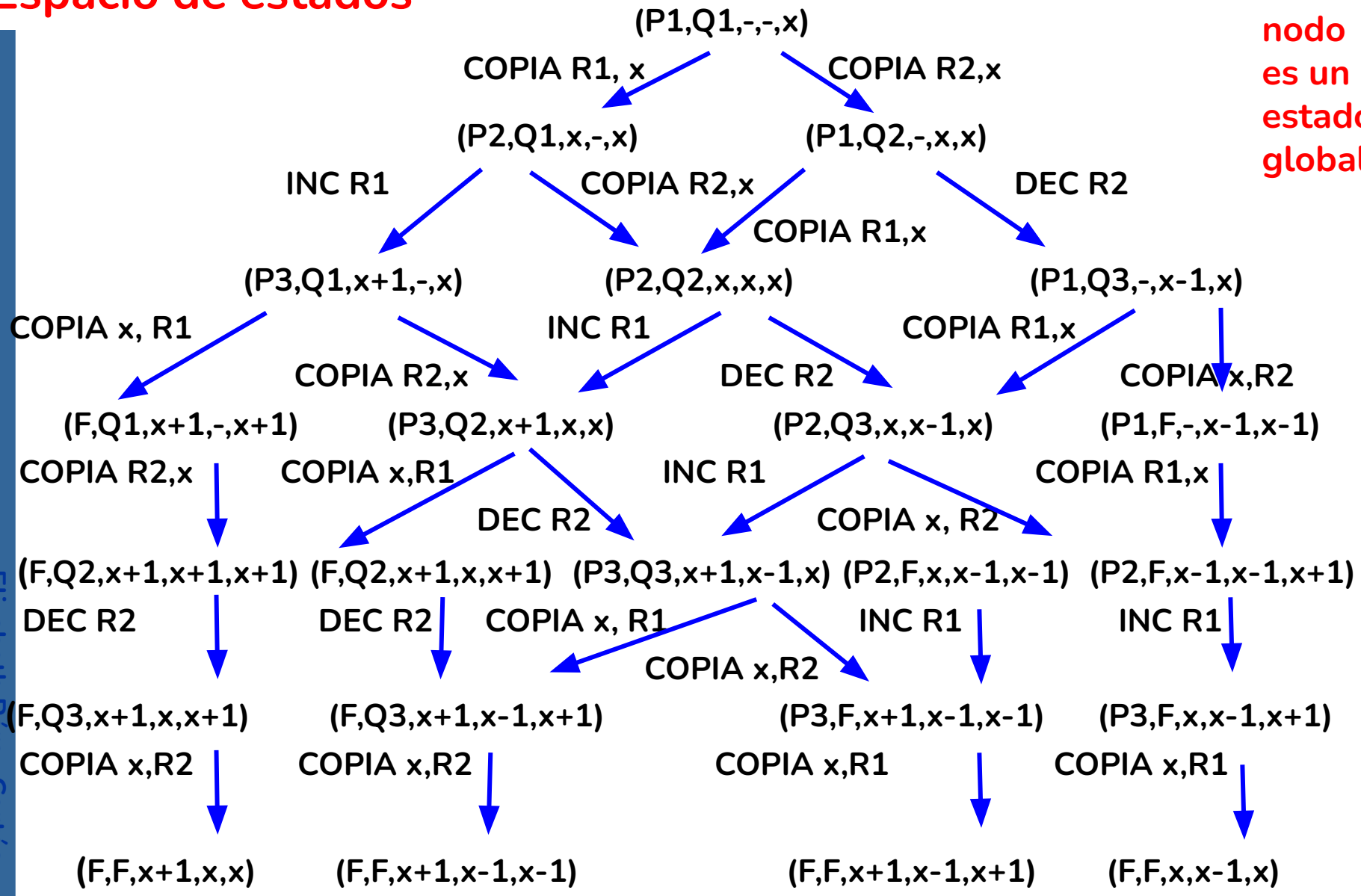
Elizabeth Pérez Cortés



Elizabeth Pérez Cortés

Espacio de estados

Cada nodo es un estado global



Cada camino desde la raíz hasta una hoja es una ejecución posible
¿cuáles son correctas?

Elizabeth Pérez Cortés

1.1 Conceptos básicos: Naturaleza no determinística de los algoritmos concurrentes

- La ejecución de un algoritmo concurrente puede resultar en la ejecución de cualquier entrelazamiento de operaciones que no viole el orden secuencial de los algoritmos que lo componen.
- Aún cuando el programa comience con los mismos datos de entrada, no siempre resulta en el mismo orden entre las operaciones de los algoritmos secuenciales que lo componen y puede no dar el mismo resultado.
- A esta característica se le conoce como no determinismo.

1.1 Conceptos básicos: Naturaleza no determinística de los algoritmos concurrentes

- En el ejemplo, la ejecución concurrente de P y Q puede dar como resultado cualquiera de los estados en el último nivel. Dos de ellos son incorrectos.
- ¿Por qué es posible terminar en un estado incorrecto?
- Esto se debe a que, en esas ejecuciones, el acceso a la variable compartida x no es exclusivo y se genera un conflicto que debemos resolver.
- Se dice que se presenta una condición de competencia sobre un recurso compartido y si al menos uno de los algoritmos secuenciales lo modifica.

Elizabeth Pérez Cortés

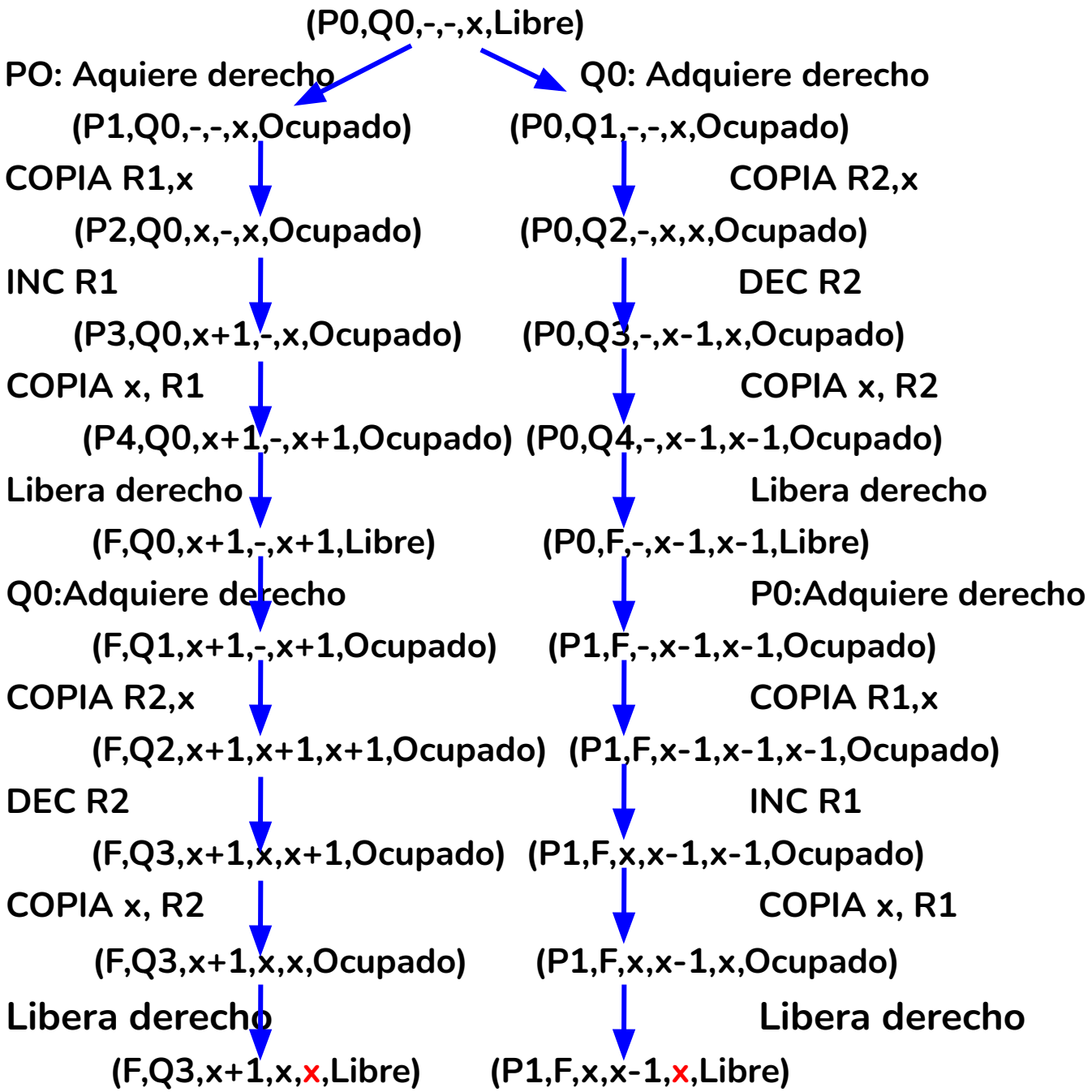
1.1 Conceptos básicos: Naturaleza no determinística de los algoritmos concurrentes

- En este caso tenemos una condición de competencia NO resuelta sobre x.
- Para resolverla, se debe sincronizar correctamente el programa marcando el uso del recurso compartido como sección crítica y se debe de seguir el protocolo para utilizar x en exclusión mutua.

```
P0: Adquiere derecho de acceso  Q0:Adquiere derecho de acceso
P1:    COPIA R1, x              Q1:    COPIA R2, x
P2:    INC R1                   Q2:    DEC R2
P3:    COPIA x, R1              Q3:    COPIA x, R2
P4: Libera derecho de acceso    Q4: Libera derecho de acceso
```

- Veamos ahora cuales son los posibles estados finales:

Elizabeth Pérez Cortés



Elizabeth Pérez Cortés

¿Cómo se demuestra que un algoritmo concurrente es correcto?

a) Es necesario demostrar que el algoritmo hace lo que tiene que hacer. Esto se hace enunciando propiedades.

b) El problema mismo determina cuales son las propiedades que el algoritmo debe tener

c) Las propiedades a probar se pueden enunciar como:

   1. Propiedades de seguridad (*safety*)
   2. Propiedades de vivacidad (*liveness*)

Elizabeth Pérez Cortés

Propiedades de seguridad (safety)

Propiedades que deben cumplirse para cada ejecución E del sistema en cualquier estado global s alcanzado en tal ejecución.

Informalmente especifican que:

NADA MALO SUCEDE NUNCA

Propiedades de seguridad (safety)

Para cada propiedad de seguridad $\mathcal{P}s$, $\mathcal{P}s$ *debe probarse verdadera en cada estado alcanzado en cada ejecución posible del algoritmo.*

Formalmente:

$$\forall \text{ ejecución } E\ (\forall \text{ estado } s \in E,\ \mathcal{P}s(s))$$

Elizabeth Pérez Cortés

Propiedades de Seguridad

EJEMPLOS:

- Nunca hay dos trenes sobre el mismo tramo de vía
- La temperatura de reactor nunca excede 100ºC
- Ningún hospitalizado se queda sin que se le revise cada 4hrs.
- El sistema no está interbloqueado (no hay progreso)

Propiedades de vivacidad (liveness)

Propiedades P que deben cumplirse en algún estado de cada ejecución posible del sistema.

Informalmente especifican que:

LAS COSAS BUENAS OCURRIRÁN EN ALGÚN MOMENTO

Propiedades de vivacidad (liveness)

Para cada propiedad de vivacidad $\mathcal{Pv}$ se debe probar que $\mathcal{Pv}$ *será verdadera en algún momento de cada ejecución posible.*

Formalmente:

$\forall$ ejecución E ($\exists$ estado s $\in$ E tal que $\mathcal{Pv}$(s))

Elizabeth Pérez Cortés

Propiedades de vivacidad (liveness)

EJEMPLOS:

- “Si el coche está apagado el ventilador se apagará”
- “Si hay una pérdida de presión, las mascarillas de oxígeno bajan”
- “El programa terminará” (no hay interbloqueos)
- “Si un proceso solicita un recurso, se le asignará en algún momento” (no hay inanición)

Algoritmo concurrente correcto

Las propiedades a probar pueden tener una naturaleza dual:

$P_{Vivacidad}$: “El programa termina”

$P_{Seguridad}$: “El programa no se interbloquea”

¿Cómo se demuestra que un algoritmo es correcto?

a) El problema mismo determina cuales son las propiedades que el algoritmo debe cumplir.
b) Se identifican las propiedades
c) Se demuestra que:
   - Para cada propiedad de seguridad $\mathcal{P}s$

   $\forall$ ejecución E ($\forall$ estado s $\in$ E $\mathcal{P}s$(s))
   - Para cada propiedad de vivacidad $\mathcal{P}v$

   $\forall$ ejecución E ($\exists$ estado s $\in$ E $\mathcal{P}v$(s))

Elizabeth Pérez Cortés

¿Cómo se demuestra que un algoritmo es correcto?

Note que demostrar lo anterior implicaría

- construir el espacio de estados
- Ver que en todos los estados se cumplen las propiedades de seguridad
- Ver que en cada camino existe un estado en el que se cumple la propiedad de vivacidad

Esta labor no puede ser hecha de manera manual así que hay todo un campo de las ciencias de la computación llamado Verificación formal de programas que se produce herramientas teóricas y tecnológicas para automatizar esta tarea.

- En general, en este tipo de programación, se hace el mejor esfuerzo.