
# Paradigmas Fundamentales de Programación

## Introducción al modelo declarativo

Juan Francisco Díaz Frias

Maestría en Ingeniería, Énfasis en Ingeniería de Sistemas y Computación
Escuela de Ingeniería de Sistemas y Computación,
home page: http://eisc.univalle.edu.co
Universidad del Valle - Cali, Colombia

# Modelos y Paradigmas de Programación

Universidad del Valle

## Plan

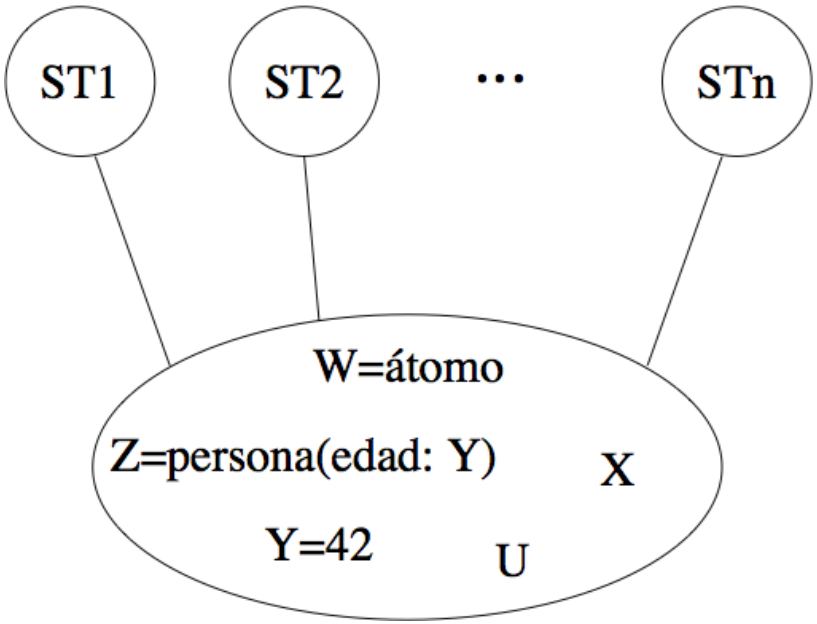
# Modelos y Paradigmas de Programación

Universidad del Valle

## Plan

# Plan

# El modelo concurrente dirigido por los datos (1)




Múltiples pilas semánticas (‘‘hilos’’)

Almacén de asignación única

Universidad del Valle

# El modelo concurrente dirigido por los datos (2)

| | |
|---|---|
| $\langle d \rangle$ ::= | |
| **skip** | Declaración vacía |
| $\langle d \rangle_1$ $\langle d \rangle_2$ | Declaración de secuencia |
| **local** $\langle x \rangle$ **in** $\langle d \rangle$ **end** | Creación de variable |
| $\langle x \rangle_1 = \langle x \rangle_2$ | Ligadura variable-variable |
| $\langle x \rangle = \langle v \rangle$ | Creación de valor |
| **if** $\langle x \rangle$ **then** $\langle d \rangle_1$ **else** $\langle d \rangle_2$ **end** | Condicional |
| **case** $\langle x \rangle$ **of** $\langle$patrón$\rangle$ **then** $\langle d \rangle_1$ **else** $\langle d \rangle_2$ **end** | Reconocimiento de patrones |
| $\{ \langle x \rangle \langle y \rangle_1 \cdots \langle y \rangle_n \}$ | Invocación de procedimiento |
| **thread** $\langle d \rangle$ **end** | Creación de hilo |

# Plan

# El modelo concurrente dirigido por los datos (3)

## Intercalación

¿Qué significa “al mismo tiempo.”?

- Punto de vista del lenguaje: los hilos se ejecutan de manera intercalada; hay una secuencia global de etapas de computación y los hilos van tomando su turno para realizar etapas de computación.
- Punto de vista de la implementación: ¿Cómo se implementan los múltiples hilos en una máquina real?
  - Un solo procesador: intercalación.
  - En el curso: semántica de intercalación. Cualquiera que sea la ejecución paralela, siempre existe al menos una intercalación observacionalmente equivalente a ella.

# El modelo concurrente dirigido por los datos (3)

## Intercalación

¿Qué significa “al mismo tiempo.”?

- Punto de vista del lenguaje: los hilos se ejecutan de manera intercalada; hay una secuencia global de etapas de computación y los hilos van tomando su turno para realizar etapas de computación.
- Punto de vista de la implementación: ¿Cómo se implementan los múltiples hilos en una máquina real?
  - Un solo procesador: intercalación.
  - Múltiples procesadores: paralelismo + intercalación.
- En el curso: semántica de intercalación. Cualquiera que sea la ejecución paralela, siempre existe al menos una intercalación observacionalmente equivalente a ella.

# El modelo concurrente dirigido por los datos (3)

## Intercalación

¿Qué significa “al mismo tiempo.”?

- Punto de vista del lenguaje: los hilos se ejecutan de manera intercalada; hay una secuencia global de etapas de computación y los hilos van tomando su turno para realizar etapas de computación.
- Punto de vista de la implementación: ¿Cómo se implementan los múltiples hilos en una máquina real?
  - Un solo procesador: intercalación.
  - Múltiples procesadores: paralelismo + intercalación.
- En el curso: semántica de intercalación. Cualquiera que sea la ejecución paralela, siempre existe al menos una intercalación observacionalmente equivalente a ella.

# El modelo concurrente dirigido por los datos (3)

## Intercalación

¿Qué significa “al mismo tiempo.”?

- Punto de vista del lenguaje: los hilos se ejecutan de manera intercalada; hay una secuencia global de etapas de computación y los hilos van tomando su turno para realizar etapas de computación.
- Punto de vista de la implementación: ¿Cómo se implementan los múltiples hilos en una máquina real?
  - Un solo procesador: intercalación.
  - Múltiples procesadores: paralelismo + intercalación.
- En el curso: semántica de intercalación. Cualquiera que sea la ejecución paralela, siempre existe al menos una intercalación observacionalmente equivalente a ella.

# El modelo concurrente dirigido por los datos (3)

## Intercalación

¿Qué significa “al mismo tiempo.”?

- Punto de vista del lenguaje: los hilos se ejecutan de manera intercalada; hay una secuencia global de etapas de computación y los hilos van tomando su turno para realizar etapas de computación.
- Punto de vista de la implementación: ¿Cómo se implementan los múltiples hilos en una máquina real?
  - Un solo procesador: intercalación.
  - Múltiples procesadores: paralelismo + intercalación.
- En el curso: semántica de intercalación. Cualquiera que sea la ejecución paralela, siempre existe al menos una intercalación observacionalmente equivalente a ella.

# El modelo concurrente dirigido por los datos (3)

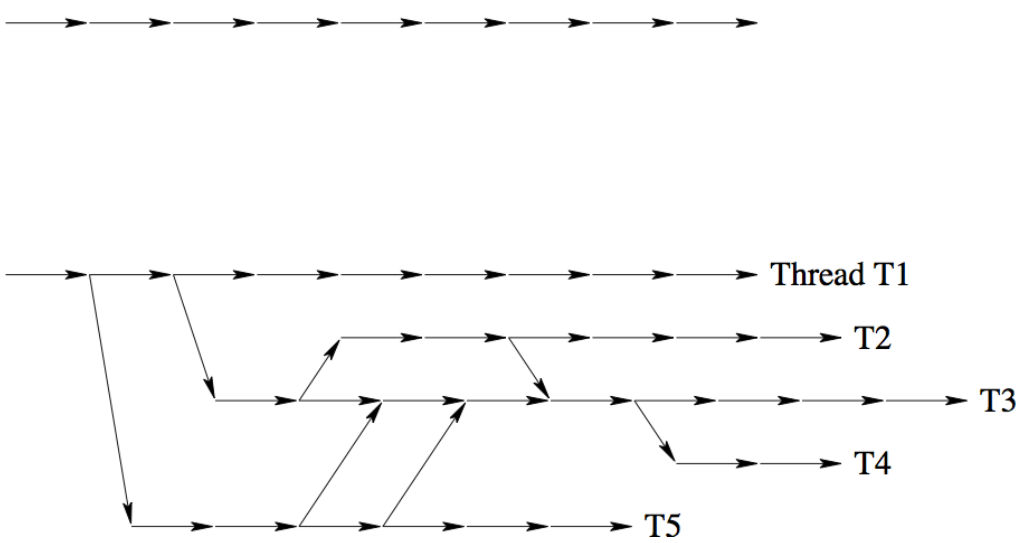
## Intercalación

¿Qué significa “al mismo tiempo.”?

- Punto de vista del lenguaje: los hilos se ejecutan de manera intercalada; hay una secuencia global de etapas de computación y los hilos van tomando su turno para realizar etapas de computación.
- Punto de vista de la implementación: ¿Cómo se implementan los múltiples hilos en una máquina real?
  - Un solo procesador: intercalación.
  - Múltiples procesadores: paralelismo + intercalación.
- En el curso: semántica de intercalación. Cualquiera que sea la ejecución paralela, siempre existe al menos una intercalación observacionalmente equivalente a ella.

# El modelo concurrente dirigido por los datos (4)

## Orden causal




Ejecución secuencial
(orden total)

etapa de computación

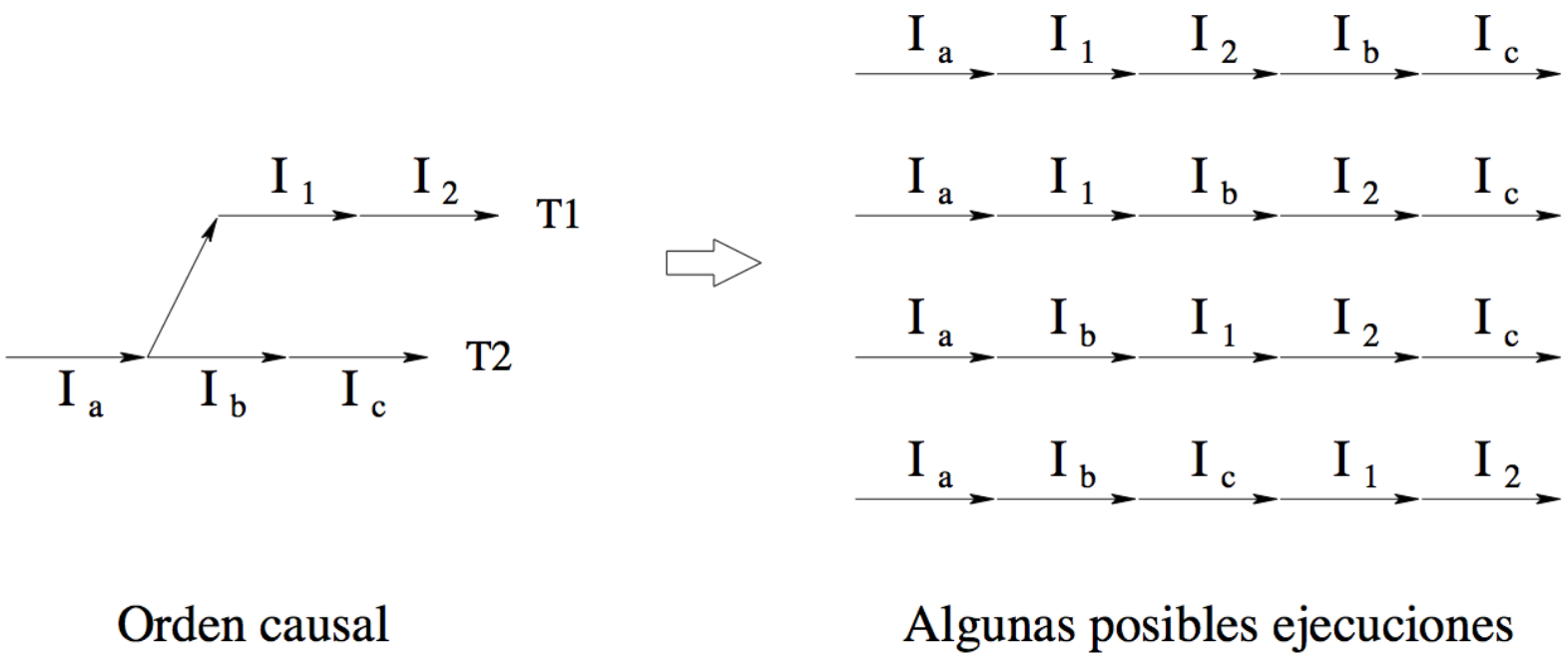
Ejejcución concurrente
(orden parcial)

orden dentro de un hilo

orden entre hilos

# El modelo concurrente dirigido por los datos (5)

## Orden causal y ejecuciones intercaladas




Orden causal

Algunas posibles ejecuciones

# Modelos y Paradigmas de Programación

Universidad del Valle

## Plan

# El modelo concurrente dirigido por los datos (6)

## No-determinismo

- Aparece naturalmente cuando hay estados concurrentes.
- Si hay varios hilos, el sistema tiene que escoger cuál hilo ejecutar en el siguiente paso.
- En un modelo concurrente declarativo, el no-determinismo no es visible al programador:
  - Las variables de flujo de datos sólo pueden ser ligadas a un valor. El no-determinismo afecta sólo el momento exacto en el que la ligadura debe tener lugar, no afecta el hecho de que la ligadura tenga lugar.
- Como consecuencia, un modelo declarativo concurrente preserva las propiedades buenas del modelo declarativo.

# El modelo concurrente dirigido por los datos (6)

## No-determinismo

- Aparece naturalmente cuando hay estados concurrentes.
- Si hay varios hilos, el sistema tiene que escoger cuál hilo ejecutar en el siguiente paso.
- En un modelo concurrente declarativo, el no-determinismo no es visible al programador:
  - Las variables de flujo de datos sólo pueden ser ligadas a un valor. El no-determinismo afecta sólo el momento exacto en que cada ligadura tiene lugar; no afecta el hecho de que la ligadura tenga lugar.
  - Cualquier operación que necesite el valor de una variable no tiene salida diferente a esperar hasta que la variable sea ligada.
- Como consecuencia, un modelo declarativo concurrente preserva las propiedades buenas del modelo declarativo.

# El modelo concurrente dirigido por los datos (6)

## No-determinismo

- Aparece naturalmente cuando hay estados concurrentes.
- Si hay varios hilos, el sistema tiene que escoger cuál hilo ejecutar en el siguiente paso.
- En un modelo concurrente declarativo, el no-determinismo no es visible al programador:
  - Las variables de flujo de datos sólo pueden ser ligadas a un valor. El no-determinismo afecta sólo el momento exacto en que cada ligadura tiene lugar; no afecta el hecho de que la ligadura tenga lugar.
  - Cualquier operación que necesite el valor de una variable no tiene salida diferente a esperar hasta que la variable sea ligada.
- Como consecuencia, un modelo declarativo concurrente preserva las propiedades buenas del modelo declarativo.

# El modelo concurrente dirigido por los datos (6)

## No-determinismo

- Aparece naturalmente cuando hay estados concurrentes.
- Si hay varios hilos, el sistema tiene que escoger cuál hilo ejecutar en el siguiente paso.
- En un modelo concurrente declarativo, el no-determinismo no es visible al programador:
  - Las variables de flujo de datos sólo pueden ser ligadas a un valor. El no-determinismo afecta sólo el momento exacto en que cada ligadura tiene lugar; no afecta el hecho de que la ligadura tenga lugar.
  - Cualquier operación que necesite el valor de una variable no tiene salida diferente a esperar hasta que la variable sea ligada.
- Como consecuencia, un modelo declarativo concurrente preserva las propiedades buenas del modelo declarativo.

# El modelo concurrente dirigido por los datos (6)

## No-determinismo

- Aparece naturalmente cuando hay estados concurrentes.
- Si hay varios hilos, el sistema tiene que escoger cuál hilo ejecutar en el siguiente paso.
- En un modelo concurrente declarativo, el no-determinismo no es visible al programador:
  - Las variables de flujo de datos sólo pueden ser ligadas a un valor. El no-determinismo afecta sólo el momento exacto en que cada ligadura tiene lugar; no afecta el hecho de que la ligadura tenga lugar.
  - Cualquier operación que necesite el valor de una variable no tiene salida diferente a esperar hasta que la variable sea ligada.
- Como consecuencia, un modelo declarativo concurrente preserva las propiedades buenas del modelo declarativo.

# El modelo concurrente dirigido por los datos (6)

## No-determinismo

- Aparece naturalmente cuando hay estados concurrentes.
- Si hay varios hilos, el sistema tiene que escoger cuál hilo ejecutar en el siguiente paso.
- En un modelo concurrente declarativo, el no-determinismo no es visible al programador:
  - Las variables de flujo de datos sólo pueden ser ligadas a un valor. El no-determinismo afecta sólo el momento exacto en que cada ligadura tiene lugar; no afecta el hecho de que la ligadura tenga lugar.
  - Cualquier operación que necesite el valor de una variable no tiene salida diferente a esperar hasta que la variable sea ligada.
- Como consecuencia, un modelo declarativo concurrente preserva las propiedades buenas del modelo declarativo.

# El modelo concurrente dirigido por los datos (7)

## Planificación

- La decisión de cuál hilo ejecutar en el próximo paso es realizada por el planificador.
- En cada etapa de computación, el planificador escoge uno de todos los hilos ejcutables, para ejecutarlo en el siguiente paso.
- Se dice que un hilo es ejecutable (o está listo), si su declaración cuenta con toda la información que necesita para ejecutar al menos una etapa de computación.
- Un hilo ejecutable se puede ejecutar en cualquier momento.
- Un hilo que no está listo se dice que está suspendido. Su primera declaración está bloqueada.
- El sistema es imparcial si no deja a ningún hilo listo "con hambre"; i.e., todos los hilos listos se ejecutan finalmente.
- La imparcialidad implica que la ejecución de un hilo no depende de la ejecución de ningún otro hilo, a menos que la dependencia sea programada explícitamente.

# El modelo concurrente dirigido por los datos (7)

## Planificación

- La decisión de cuál hilo ejecutar en el próximo paso es realizada por el planificador.
- En cada etapa de computación, el planificador escoge uno de todos los hilos ejcutables, para ejecutarlo en el siguiente paso.
- Se dice que un hilo es ejecutable (o está listo), si su declaración cuenta con toda la información que necesita para ejecutar al menos una etapa de computación.
- Un hilo ejecutable se puede ejecutar en cualquier momento.
- Un hilo que no está listo se dice que está suspendido. Su primera declaración está bloqueada.
- El sistema es imparcial si no deja a ningún hilo listo "con hambre"; i.e., todos los hilos listos se ejecutan finalmente.
- La imparcialidad implica que la ejecución de un hilo no depende de la ejecución de ningún otro hilo, a menos que la dependencia sea programada explícitamente.

# El modelo concurrente dirigido por los datos (7)

## Planificación

- La decisión de cuál hilo ejecutar en el próximo paso es realizada por el planificador.
- En cada etapa de computación, el planificador escoge uno de todos los hilos ejcutables, para ejecutarlo en el siguiente paso.
- Se dice que un hilo es ejecutable (o está listo), si su declaración cuenta con toda la información que necesita para ejecutar al menos una etapa de computación.
- Un hilo ejecutable se puede ejecutar en cualquier momento.
- Un hilo que no está listo se dice que está suspendido. Su primera declaración está bloqueada.
- El sistema es imparcial si no deja a ningún hilo listo "con hambre"; i.e., todos los hilos listos se ejecutan finalmente.
- La imparcialidad implica que la ejecución de un hilo no depende de la ejecución de ningún otro hilo, a menos que la dependencia sea programada explícitamente.

# El modelo concurrente dirigido por los datos (7)

## Planificación

- La decisión de cuál hilo ejecutar en el próximo paso es realizada por el planificador.
- En cada etapa de computación, el planificador escoge uno de todos los hilos ejcutables, para ejecutarlo en el siguiente paso.
- Se dice que un hilo es ejecutable (o está listo), si su declaración cuenta con toda la información que necesita para ejecutar al menos una etapa de computación.
- Un hilo ejecutable se puede ejecutar en cualquier momento.
- Un hilo que no está listo se dice que está suspendido. Su primera declaración está bloqueada.
- El sistema es imparcial si no deja a ningún hilo listo "con hambre"; i.e., todos los hilos listos se ejecutan finalmente.
- La imparcialidad implica que la ejecución de un hilo no depende de la ejecución de ningún otro hilo, a menos que la dependencia sea programada explícitamente.

# El modelo concurrente dirigido por los datos (7)

## Planificación

- La decisión de cuál hilo ejecutar en el próximo paso es realizada por el planificador.
- En cada etapa de computación, el planificador escoge uno de todos los hilos ejcutables, para ejecutarlo en el siguiente paso.
- Se dice que un hilo es ejecutable (o está listo), si su declaración cuenta con toda la información que necesita para ejecutar al menos una etapa de computación.
- Un hilo ejecutable se puede ejecutar en cualquier momento.
- Un hilo que no está listo se dice que está suspendido. Su primera declaración está bloqueada.
- El sistema es imparcial si no deja a ningún hilo listo “con hambre”; i.e., todos los hilos listos se ejecutan finalmente.
- La imparcialidad implica que la ejecución de un hilo no depende de la ejecución de ningún otro hilo, a menos que la dependencia sea programada explícitamente.

# El modelo concurrente dirigido por los datos (7)

## Planificación

- La decisión de cuál hilo ejecutar en el próximo paso es realizada por el planificador.
- En cada etapa de computación, el planificador escoge uno de todos los hilos ejcutables, para ejecutarlo en el siguiente paso.
- Se dice que un hilo es ejecutable (o está listo), si su declaración cuenta con toda la información que necesita para ejecutar al menos una etapa de computación.
- Un hilo ejecutable se puede ejecutar en cualquier momento.
- Un hilo que no está listo se dice que está suspendido. Su primera declaración está bloqueada.
- El sistema es imparcial si no deja a ningún hilo listo “con hambre”; i.e., todos los hilos listos se ejecutan finalmente.
- La imparcialidad implica que la ejecución de un hilo no depende de la ejecución de ningún otro hilo, a menos que la dependencia sea programada explícitamente.

# El modelo concurrente dirigido por los datos (7)

## Planificación

- La decisión de cuál hilo ejecutar en el próximo paso es realizada por el planificador.
- En cada etapa de computación, el planificador escoge uno de todos los hilos ejcutables, para ejecutarlo en el siguiente paso.
- Se dice que un hilo es ejecutable (o está listo), si su declaración cuenta con toda la información que necesita para ejecutar al menos una etapa de computación.
- Un hilo ejecutable se puede ejecutar en cualquier momento.
- Un hilo que no está listo se dice que está suspendido. Su primera declaración está bloqueada.
- El sistema es imparcial si no deja a ningún hilo listo “con hambre”; i.e., todos los hilos listos se ejecutan finalmente.
- La imparcialidad implica que la ejecución de un hilo no depende de la ejecución de ningún otro hilo, a menos que la dependencia sea programada explícitamente.

# Plan

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle d \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$(\underbrace{\{\underbrace{[\underbrace{(\langle d \rangle, \emptyset)}_{\text{declaración}}]}_{\text{pila}}\}}_{\text{multiconjunto}}, \emptyset)$$

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle d \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$(\{\underbrace{[\underbrace{(\langle d \rangle, \emptyset)}_{\text{declaración}}]}_{\text{pila}}\}, \emptyset)$$

multiconjunto

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle \mathrm{d} \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$\underbrace{(\{\underbrace{[\overbrace{(\langle \mathrm{d} \rangle, \emptyset)}^{\text{declaración}}]}_{\text{pila}}\}}_{\text{multiconjunto}}, \emptyset)$$

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle \mathrm{d} \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$(\{\underbrace{[\overbrace{(\langle \mathrm{d} \rangle, \emptyset)}^{\text{declaración}}]}_{\text{pila}}\}, \emptyset)$$

multiconjunto

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle \mathrm{d} \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$(\{\underbrace{[\overbrace{(\langle \mathrm{d} \rangle, \emptyset)}^{\text{declaración}}]}_{\text{pila}}\}, \emptyset)$$

multiconjunto

## Ejecución concurrente

- Etapa de computación: $(\{ST\} \uplus MST', \sigma) \rightarrow (\{ST'\} \uplus MST', \sigma')$
  Semántica de intercalación
- La elección de cuál $ST$ seleccionar es realizada por el planificador: debe asegurar imparcialidad.
- terminación: No hay pilas semánticas ejecutables en $MST$ y cada $ST$ de $MST$ está terminado.
- bloqueo: No hay pilas semánticas ejecutables en $MST$ y existe por lo menos un hilo, $ST$ de $MST$, suspendido, que no puede ser recuperado.

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle \mathrm{d} \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$(\{\underbrace{[\overbrace{(\langle \mathrm{d} \rangle, \emptyset)}^{\text{declaración}}]}_{\text{pila}}\}, \emptyset)$$

multiconjunto

## Ejecución concurrente

- Etapa de computación: $(\{ST\} \uplus MST', \sigma) \rightarrow (\{ST'\} \uplus MST', \sigma')$
  Semántica de intercalación
- La elección de cuál $ST$ seleccionar es realizada por el planificador: debe asegurar imparcialidad.
- terminación: No hay pilas semánticas ejecutables en $MST$ y cada $ST$ de $MST$ está terminado.
- bloqueo: No hay pilas semánticas ejecutables en $MST$ y existe por lo menos un hilo, $ST$ de $MST$, suspendido, que no puede ser recuperado.

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle \mathrm{d} \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$(\{\underbrace{[\overbrace{(\langle \mathrm{d} \rangle, \emptyset)}^{\text{declaración}}]}_{\text{pila}}\}, \emptyset)$$

multiconjunto

## Ejecución concurrente

- Etapa de computación: $(\{ST\} \uplus MST', \sigma) \rightarrow (\{ST'\} \uplus MST', \sigma')$
  Semántica de intercalación
- La elección de cuál $ST$ seleccionar es realizada por el planificador: debe asegurar imparcialidad.
- terminación: No hay pilas semánticas ejecutables en $MST$ y cada $ST$ de $MST$ está terminado.
- bloqueo: No hay pilas semánticas ejecutables en $MST$ y existe por lo menos un hilo, $ST$ de $MST$, suspendido, que no puede ser recuperado.

# El modelo concurrente dirigido por los datos (8)

## Semántica de los hilos

Conservamos los conceptos de almacén de asignación única $\sigma$, ambiente $E$, declaración semántica $(\langle \mathrm{d} \rangle, E)$, y pila semántica $ST$.

- Un estado de ejecución es una pareja $(MST, \sigma)$ donde $MST$ es un multiconjunto de pilas semánticas y $\sigma$ es un almacén de asignación única.
- Una computación es una secuencia de estados de ejecución: $(MST_0, \sigma_0) \rightarrow (MST_1, \sigma_1) \rightarrow (MST_2, \sigma_2) \rightarrow \cdots$.

## Ejecución concurrente

Estado inicial:

$$(\{\underbrace{[\overbrace{(\langle \mathrm{d} \rangle, \emptyset)}^{\text{declaración}}]}_{\text{pila}}\}, \emptyset)$$

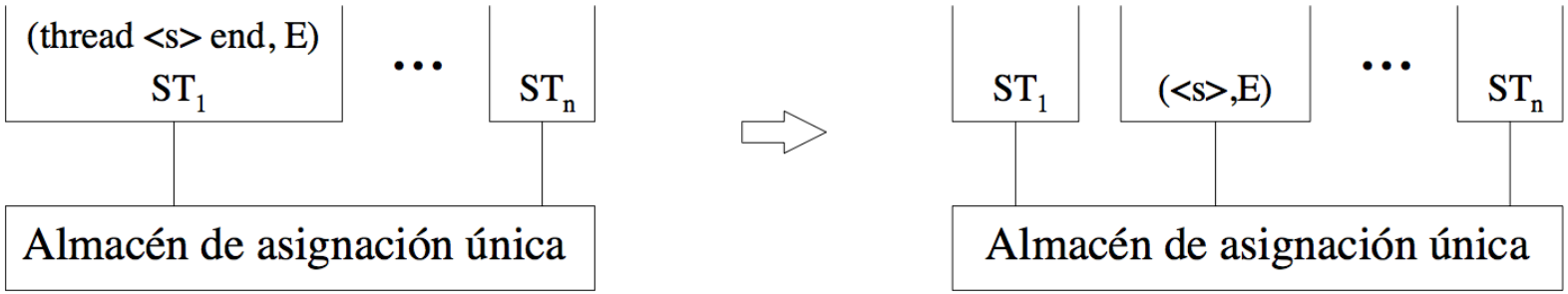
multiconjunto

## Ejecución concurrente

- Etapa de computación: $(\{ST\} \uplus MST', \sigma) \rightarrow (\{ST'\} \uplus MST', \sigma')$
  Semántica de intercalación
- La elección de cuál $ST$ seleccionar es realizada por el planificador: debe asegurar imparcialidad.
- terminación: No hay pilas semánticas ejecutables en $MST$ y cada $ST$ de $MST$ está terminado.
- bloqueo: No hay pilas semánticas ejecutables en $MST$ y existe por lo menos un hilo, $ST$ de $MST$, suspendido, que no puede ser recuperado.

# El modelo concurrente dirigido por los datos (9)

## Semántica de `thread`

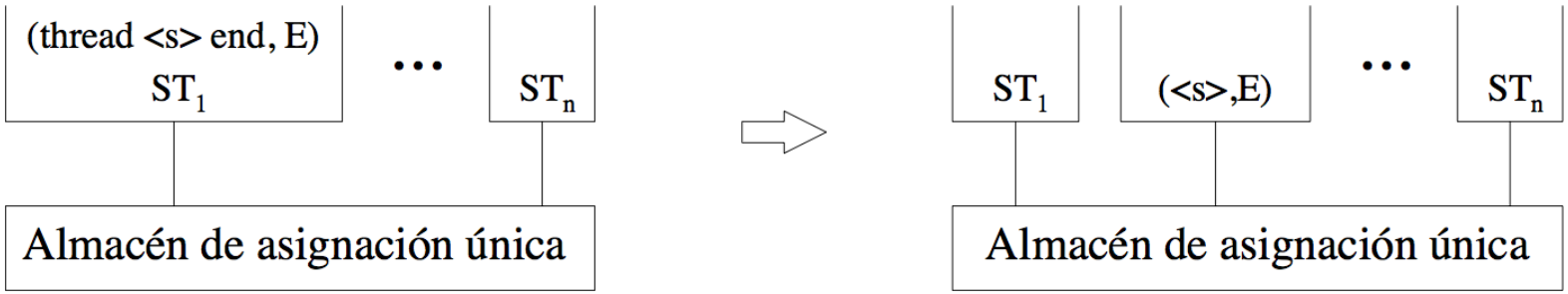



## Administración de la memoria

- Una pila semántica terminada puede ser desalojada.
- Una pila semántica bloqueada puede ser recuperada como memoria libre si su condición de activación depende de una variable inalcanzable. En ese caso, la pila semántica nunca se volverá ejecutable de nuevo, y por ello eliminarla no cambia nada durante la ejecución.

# El modelo concurrente dirigido por los datos (9)

## Semántica de `thread`




## Administración de la memoria

- Una pila semántica terminada puede ser desalojada.
- Una pila semántica bloqueada puede ser recuperada como memoria libre si su condición de activación depende de una variable inalcanzable. En ese caso, la pila semántica nunca se volverá ejecutable de nuevo, y por ello eliminarla no cambia nada durante la ejecución.

# El modelo concurrente dirigido por los datos (10)

## Ejemplo

```
local B in
   thread
     B=true
   end
   if B then
     {Browse si}
   end
end
```

# El modelo concurrente dirigido por los datos (10)

## Ejemplo

```
local B in
   thread
     B=true
   end
   if B then
     {Browse si}
   end
end
```

## Ejecución

- Después de las etapas iniciales:
  $(\{[\texttt{thread}\ b\texttt{=true end},\ \texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
- Después de ejecutar `thread`:
  $(\{[b\texttt{=true}],\ [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
  Un hilo queda supendido. ¿Por qué?
- Turno para el hilo listo:
  $(\{[],\ [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Ahora sí, ejecución del `if`:
  $(\{[\{\texttt{Browse si}\}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Y finalmente se ejecuta el `Browse`.

# El modelo concurrente dirigido por los datos (10)

## Ejemplo

```
local B in
   thread
     B=true
   end
   if B then
     {Browse si}
   end
end
```

## Ejecución

- Después de las etapas iniciales:
  $(\{[\texttt{thread}\ b\texttt{=true}\ \texttt{end},\ \texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
- Después de ejecutar **thread**:
  $(\{[b\texttt{=true}], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
  Un hilo queda supendido. ¿Por qué?
- Turno para el hilo listo:
  $(\{[], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Ahora sí, ejecución del **if**:
  $(\{[\{\texttt{Browse si}\}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Y finalmente se ejecuta el Browse.

# El modelo concurrente dirigido por los datos (10)

## Ejemplo

```
local B in
   thread
     B=true
   end
   if B then
     {Browse si}
   end
end
```

## Ejecución

- Después de las etapas iniciales:
  $(\{[\texttt{thread}\ b\texttt{=true}\ \texttt{end},\ \texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
- Después de ejecutar **thread**:
  $(\{[b\texttt{=true}], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
  Un hilo queda supendido. ¿Por qué?
- Turno para el hilo listo:
  $(\{[], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Ahora sí, ejecución del **if**:
  $(\{[\{\texttt{Browse si}\}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Y finalmente se ejecuta el Browse.

# El modelo concurrente dirigido por los datos (10)

## Ejemplo

```
local B in
   thread
     B=true
   end
   if B then
     {Browse si}
   end
end
```

## Ejecución

- Después de las etapas iniciales:
  $(\{[\texttt{thread}\ b\texttt{=true}\ \texttt{end},\ \texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
- Después de ejecutar **thread**:
  $(\{[b\texttt{=true}], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
  Un hilo queda supendido. ¿Por qué?
- Turno para el hilo listo:
  $(\{[], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Ahora sí, ejecución del **if**:
  $(\{[\{\texttt{Browse si}\}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Y finalmente se ejecuta el `Browse`.

# El modelo concurrente dirigido por los datos (10)

## Ejemplo

```
local B in
   thread
     B=true
   end
   if B then
     {Browse si}
   end
end
```

## Ejecución

- Después de las etapas iniciales:
  $(\{[\texttt{thread}\ b\texttt{=true}\ \texttt{end},\ \texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
- Después de ejecutar **thread**:
  $(\{[b\texttt{=true}], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b\} \cup \sigma)$
  Un hilo queda supendido. ¿Por qué?
- Turno para el hilo listo:
  $(\{[], [\texttt{if}\ b\ \texttt{then}\ \{\texttt{Browse si}\}\ \texttt{end}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Ahora sí, ejecución del **if**:
  $(\{[\{\texttt{Browse si}\}]\},\ \{b = \texttt{true}\} \cup \sigma)$
- Y finalmente se ejecuta el `Browse`.

# Modelos y Paradigmas de Programación

Universidad del Valle

## Plan

# Técnicas básicas de programación de hilos (1)

## Comportamiento de flujo de datos

```
declare X0 X1 X2 X3 in
thread
Y0 Y1 Y2 Y3 in
   {Browse [Y0 Y1 Y2 Y3]}
   Y0=X0+1
   Y1=X1+Y0
   Y2=X2+Y1
   Y3=X3+Y2
   {Browse listo}
end
{Browse [X0 X1 X2 X3]}
```

¿Qué pasa cuando se ejecutan las siguientes declaraciones, una a la vez?

```
X0=0
X1=1
X2=2
X3=3
```

# Técnicas básicas de programación de hilos (1)

## Comportamiento de flujo de datos

```
declare X0 X1 X2 X3 in
thread
Y0 Y1 Y2 Y3 in
   {Browse [Y0 Y1 Y2 Y3]}
   Y0=X0+1
   Y1=X1+Y0
   Y2=X2+Y1
   Y3=X3+Y2
   {Browse listo}
end
{Browse [X0 X1 X2 X3]}
```

¿Qué pasa cuando se ejecutan las siguientes declaraciones, una a la vez?

```
X0=0
X1=1
X2=2
X3=3
```

Modelos y Paradigmas de Programación

Universidad del Valle

# Técnicas básicas de programación de hilos (2)

## Versión concurrente de `Map`

```
fun {MapC Xs F}
   case Xs of
        nil then nil
        [] X|Xr then
           thread {F X} end|{Map
Xr F}
   end
end
```

:Ojo:**`thread`** se usa como una expresión.

Modelos y Paradigmas de Programación

Universidad del Valle

# Técnicas básicas de programación de hilos (2)

## Versión concurrente de `Map`

```
fun {MapC Xs F}
   case Xs of
      nil then nil
      [] X|Xr then
         thread {F X} end|{Map
Xr F}
   end
end
```

:Ojo:**`thread`** se usa como una expresión.

## Exploremos el nuevo `MapC`

- Ensaye:

```
declare F Xs Ys Zs
{Browse thread {MapC Xs
                     F}
        end}
```

- Ahora ensaye:

```
Xs=1|2|Ys
fun {F X} X*X end
```

- Y ahora,

```
Ys=3|Zs
Zs=nil
```

Modelos y Paradigmas de Programación

Universidad del Valle

# Técnicas básicas de programación de hilos (2)

## Versión concurrente de `Map`

```
fun {MapC Xs F}
   case Xs of
      nil then nil
      [] X|Xr then
         thread {F X} end|{Map
Xr F}
   end
end
```

:Ojo:**`thread`** se usa como una expresión.

## Exploremos el nuevo `MapC`

- Ensaye:

```
declare F Xs Ys Zs
{Browse thread {MapC Xs
                     F}
        end}
```

- Ahora ensaye:

```
Xs=1|2|Ys
fun {F X} X*X end
```

- Y ahora,

```
Ys=3|Zs
Zs=nil
```

# Técnicas básicas de programación de hilos (2)

## Versión concurrente de `Map`

```
fun {MapC Xs F}
   case Xs of
      nil then nil
      [] X|Xr then
         thread {F X} end|{Map
Xr F}
   end
end
```

:Ojo:**`thread`** se usa como una expresión.

## Exploremos el nuevo `MapC`

- Ensaye:

```
declare F Xs Ys Zs
{Browse thread {MapC Xs
                     F}
        end}
```

- Ahora ensaye:

```
Xs=1|2|Ys
fun {F X} X*X end
```

- Y ahora,

```
Ys=3|Zs
Zs=nil
```

Modelos y Paradigmas de Programación

Universidad del Valle

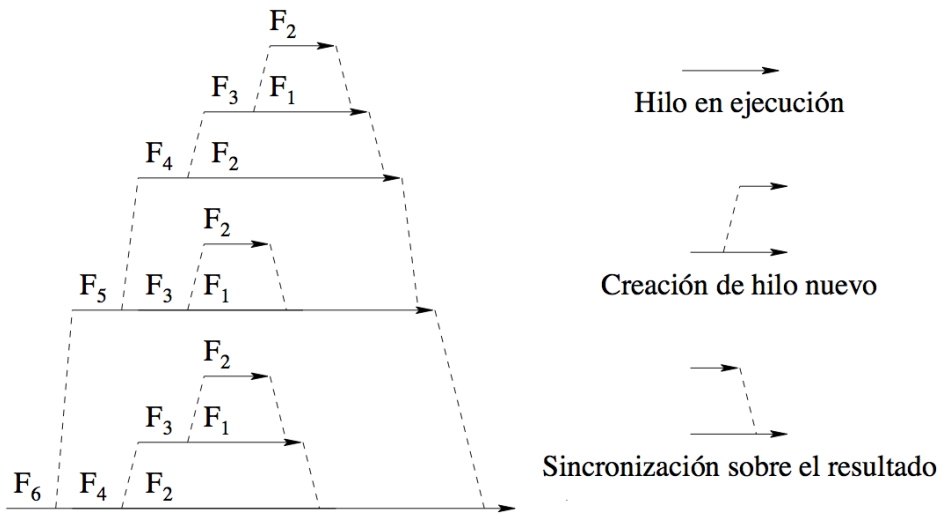
# Técnicas básicas de programación de hilos (3)

## Versión concurrente de `Fib`

```
fun {FibC X}
   if X=<2 then 1
   else thread {FibC X-1}
        end +
             {FibC X-2}
   end
end
```

- Note lo fácil que es volver concurrente un programa.
- Note el uso de `thread` como expresión.

## Concurrencia económica




Una invocación a `FibC` creará un número exponencial de hilos.
¿Cuántos puede manejar en su equipo? Mire el panel.

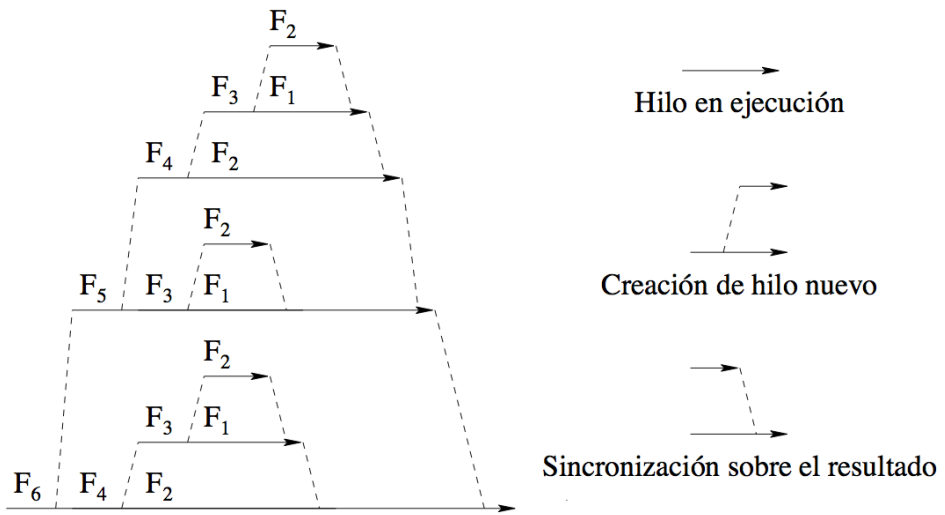
# Técnicas básicas de programación de hilos (3)

## Versión concurrente de `Fib`

```
fun {FibC X}
   if X=<2 then 1
   else thread {FibC X-1}
        end +
             {FibC X-2}
   end
end
```

- Note lo fácil que es volver concurrente un programa.
- Note el uso de **`thread`** como expresión.

## Concurrencia económica




Una invocación a `FibC` creará un número exponencial de hilos.
¿Cuántos puede manejar en su equipo? Mire el panel.

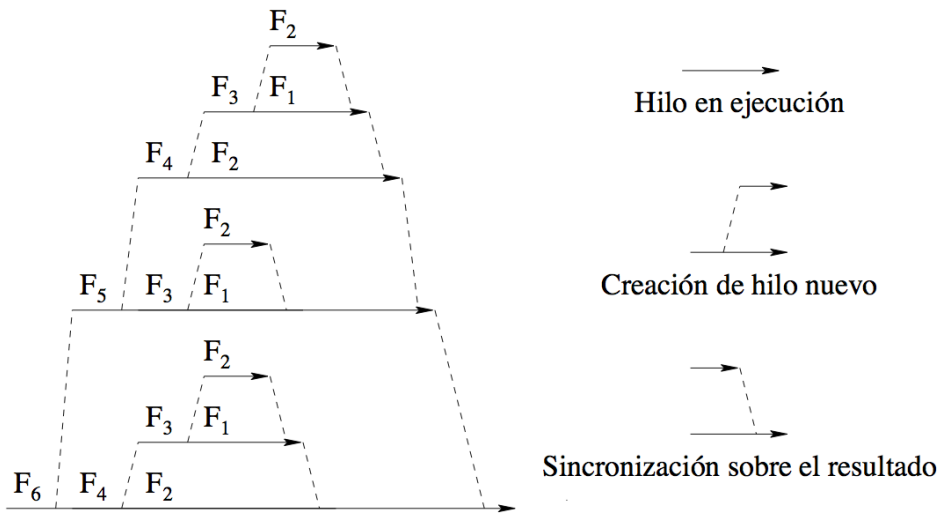
# Técnicas básicas de programación de hilos (3)

## Versión concurrente de `Fib`

```
fun {FibC X}
   if X=<2 then 1
   else thread {FibC X-1}
        end +
             {FibC X-2}
   end
end
```

- Note lo fácil que es volver concurrente un programa.
- Note el uso de **`thread`** como expresión.

## Concurrencia económica




Una invocación a `FibC` creará un número exponencial de hilos.
¿Cuántos puede manejar en su equipo? Mire el panel.