Paralelismo y Concurrencia en Sistemas

El modelo Linda

Dr. Alejandro J. García
e-mail: agarcia@cs.uns.edu.ar
http:// cs.uns.edu.ar /~ajg

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

El modelo "Linda"

- Para escribir programas con paralelismo, el lenguaje debe proveer herramientas para crear y coordinar múltiples procesos.
- Linda es un modelo de memoria que consiste de unas pocas operaciones simples, que envuelven el modelo de espacio de tuplas de la programación en paralelo.
- Agregando estas operaciones del espacio de tuplas a un lenguaje base se produce un dialecto de programación paralela que permite creación de procesos, comunicación y sincronización.

Espacio de tuplas

- El concepto principal de Linda es el de un "Espacio de tuplas", este es una abstracción via el cual los procesos cooperan y se comunican.
- Es una abstracción de un área de memoria compartida, que es referenciada de manera asociativa.
- Provee comunicación entre procesos sin requerir el hardware subyacente para la memoria compartida físicamente.

Tupla

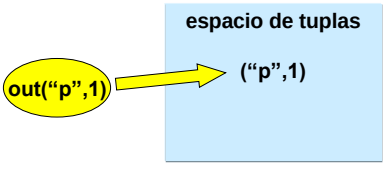
- Las tuplas son colecciones de campos agrupados para formar ítems de almacenamiento persistente.
- Cada uno de los campos es un valor de algún tipo soportado por el lenguaje base. (Linda es un modelo)
- Algunos campos son valores (o "actuals"), otros son contenedores tipados (o "formals"). Un formal lleva un signo de pregunta (?) delante.
- Por ejemplo, en la tupla ("una cadena", ?f, ?i, 5) hay dos actuals y dos formals.
- Las tuplas son direccionadas por contenido en vez de por nombre o dirección.

Primitivas del modelo Linda

- Las tuplas son la unidad de almacenamiento básico del espacio de tuplas.
- Linda sugiere cuatro operaciones básicas que manipulan el espacio de tuplas (insertando, eliminando o recuperando tuplas), y dos variantes predicativas adicionales.
- Las operaciones out y eval permiten agregar datos al espacio de tuplas (ET).
- Las operaciones rd, in, inp y rdp permiten leer o quitar elementos del ET.

out

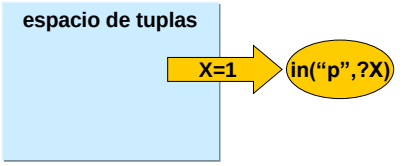
- out(T): Causa que la tupla T sea agregada al espacio de tuplas.



- Es una primitiva no bloqueante.

in

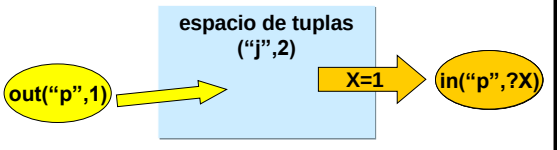
- **in(S): Causa que alguna tupla T que concuerda con la tupla-molde S sea retirada del espacio de tuplas.**



- **Es una primitiva destructiva.**
- **Si hay muchas tuplas que coinciden, se elige una arbitrariamente.**

in

- **Si no se encuentra una tupla *t* que coincida cuando in(s) se ejecuta,…**
- **la ejecución del proceso se suspende.**



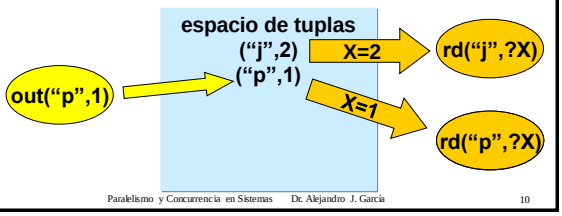
- **Permanece suspendido hasta que una tupla que coincida esté disponible.**
- **Es una primitiva bloqueante.**

in

- **El algoritmo de matching usado dependerá del lenguaje en el cual linda está embebido.**
- **Una concordancia ocurre si se reúnen las siguientes tres condiciones:**
  - **El molde y la tupla tienen el mismo número de campos.**
  - **Los tipos de los campo correspondientes son iguales.**
  - **Cada constante o variable en el molde coincide con su campo de tupla.**

rd - (read)

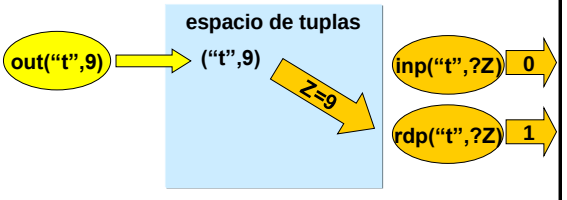
- **rd(s): es similar al *in(s)*, excepto que la tupla que coincide no es retirada y permanece en el espacio de tuplas.**
- **Es una primitiva no destructiva y bloqueante.**



inp y rdp

- **Las versiones predicativas de in y rd, inp y rdp respectivamente, intentan localizar una tupla que coincida, y retornan 0 si la búsqueda falla; en caso contrario retornan 1.**
- **inp es una primitiva destructiva y no bloqueante.**
- **rdp es una primitiva no destructiva y no bloqueante.**
- **Realizan las asignaciones de actuales a formales como se describió antes.**

inp y rdp



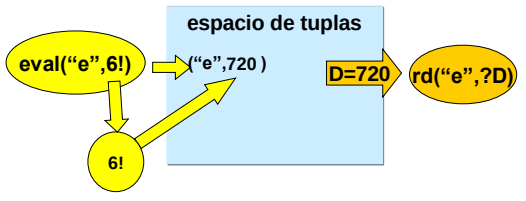
eval

- **eval(t): es similar al *out(t)*, excepto que *t* es evaluado después (en paralelo) en vez de antes de ser ingresado en el espacio de tuplas.**
- **eval crea implícitamente un nuevo proceso para evaluar cada campo de *t*. Por lo tanto:**
- **Hay dos clases de tuplas:**
  - **Tuplas de procesos**
  - **Tuplas de datos**

eval

- **Las tuplas de procesos están bajo evaluación activa.**
- **Las tuplas de datos son pasivas.**
- **Una tupla de proceso que ha finalizado su ejecución se convierte en una tupla de datos, indistinguible con respecto a otras tuplas de datos.**
- **Cuando cada campo ha sido completamente evaluado, *t* se pone como una tupla pasiva, la cual puede ser consultada por una operación *in* o *rd* como cualquier otra tupla.**

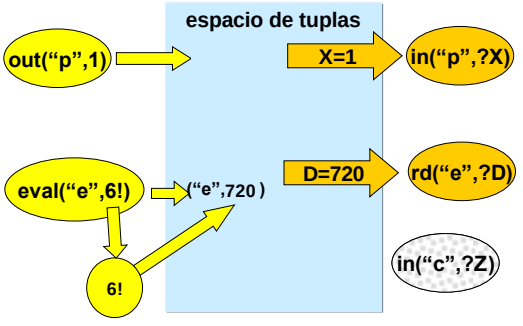
eval



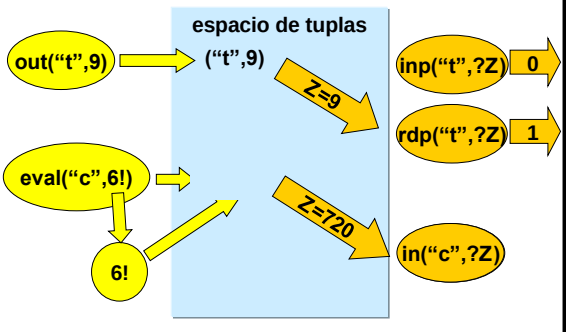
Resumen de primitivas

	out	in	rd	inp	rdp	eval
Destructiva	NO	SI	NO	SI	NO	NO
Bloqueante	NO	SI	SI	NO	NO	NO

Ejemplo: primitivas



Ejemplo: primitivas



Estructuras de datos en Linda

Workpool
- Out("pool1", descriptor)
- In("pool1", ? Tarea)

Registros (son tuplas)
- Out(juan, gomez, 23)
- In(?nombre, ?apellido, ?edad)

Arreglos y **matrices**: conjunto de tuplas
("A",1,1,componente1)
("A",1,2,componente1)

Estructuras de datos en Linda: colas

espacio de tuplas

("cola", 1, val1)
("cola", 2, val2)
("cola", 3, val3)
("cola", ultimo, 3)
("cola", pri, 1)

Operaciones de Colas:

Poner un elemento:
in("cola", ultimo,?X)
out("cola", ultimo,X+1)
out("cola", X+1,elemento)

Sacar un elemento:
in("cola", pri, ?X)
out("cola", pri, X+1)
in("cola", X, ?elemento)

FORK y FORALL en Linda

Fork:
Eval("mi-proceso",computar(…))
JOIN:
in("mi-proceso",?X)

ForALL:
For … DO
 eval("mi-bucle", computar(…))
For … DO
 in("mi-bucle", ?resultado)

Implementación de mecanismos de sincronización en Linda

Semáforos en Linda
- Wait(s)
  se implementa con in("s")
- Signal(s)
  se implementa con out("s")
- **Barreras** (caso particular)
- Una barrera para 25 procesos: out("barrera1", 25)
- Cada proceso al llegar hace:
  In("barrera1",?val); saca el valor actual
  Out("barrera1", val-1); lo agrega decrementado
  Rd("barrera1",0); espera a que sea 0
- *¿Qué ocurre si la barrera está en una repetición?*

Resumen Modelo Linda

- Agregando las operaciones del espacio de tuplas a un lenguaje base se produce un dialecto de programación paralela que permite **creación de procesos**, **comunicación** y **sincronización**.

- **Comunicación**: los procesos se comunican por medio del espacio de tuplas, agregando tuplas (out y eval), leyendo tuplas (*in*, *rd*, *inp* y *rdp*).
- **Creación de procesos**: mediante la primitiva *eval*.
- **Sincronización**: mediante las primitivas bloqueantes (*in*, *rd*).

Linda como modelo no herramienta

- Linda no es un lenguaje, ni una herramienta, es un modelo ortogonal al lenguaje de programación que lo contiene.
- Un **modelo** (o paradigma) representa una manera particular de pensar un problema. Puede ser implementado en diferentes maneras y contextos.
- Una **herramienta** de software, sin embargo, es un sistema de trabajo que puede ser usado para resolver problemas.
- Por ejemplo el sistema llamado **C-Linda**, el cual es una herramienta, es una pieza de software que soporta programación paralela.
- Desde el punto de vista de las aplicaciones, Linda es una extensión de un lenguaje de programación para facilitar la programación paralela.

El modelo "Linda"

- **Este modelo puede ser implementado en varios lenguajes de programación (de diferentes paradigmas).**
- **Linda también es usado sin importar la arquitectura subyacente.**
- **No se entromete en aspectos de computación, Linda gana libertad para coexistir con cualquier lenguaje base y modelo de computación.**
- **Soporta operaciones puras, simples y poderosas dentro de su propio dominio.**

Bibliografía

– **How to write parallel programs.**
Capítulo 3
Nicholas Carriero and David Gelernter.
1992. MIT Press.