Problemas de sincronización

Sistemas Operativos

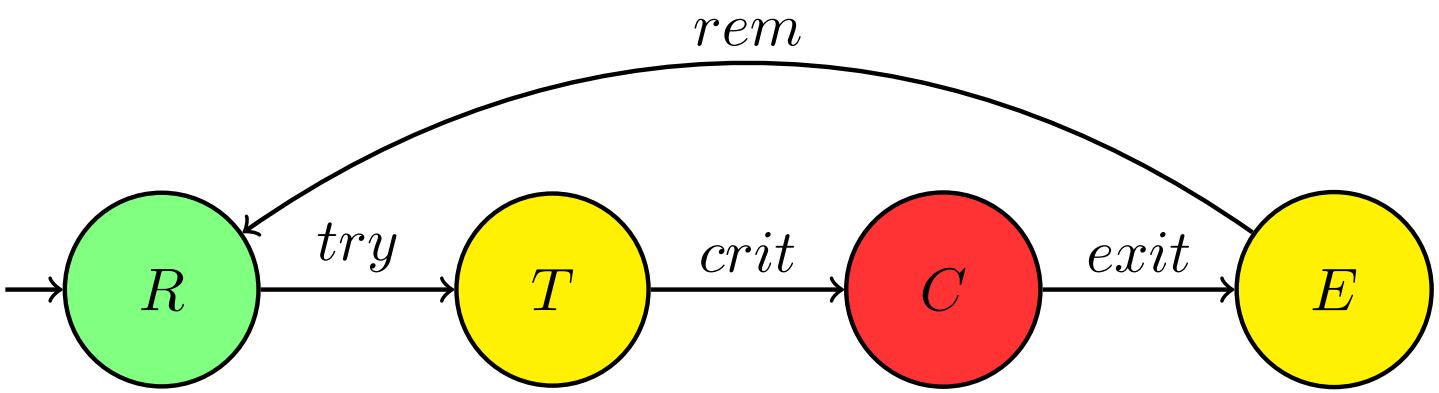
Matías Barbeito

Departamento de Computación, FCEyN, UBA

7 de septiembre de 2017

Segundo cuatrimestre de 2017

Modelo de procesos



- Estado: $\sigma : [0 \dots N-1] \mapsto \{R, T, C, E\}$
- Transición: $\sigma \xrightarrow{\ell} \sigma'$, $\ell \in \{rem, try, crit, exit\}$
- Ejecución: $\tau = \tau_0 \xrightarrow{\ell} \tau_1 \dots$
- Garantizar $PROP$: Toda ejecución satisface $PROP$
- Notación: $\#S =$ cantidad de elementos del conjunto S

Ejecuciones (o trazas)

- τ_k representa al estado del proceso en el instante k.
- Como hay varios procesos usamos $\tau_k(i)$ para el estado del proceso i en el instante k.
- Una ejecución es una secuencia de estados.
- Para hablar de las propiedades dentro de este modelo no nos alcanza la lógica de primer orden.

Enunciado de Barrera

Queremos hacer una Barrera de sincronización o `Rendezvous` (punto de encuentro). Cada $P_i, i \in [0 \dots N-1]$, tiene que ejecutar $a(i); b(i)$, pero sólo puede ejecutar $b(i)$ si para todo $j \in [0 \dots N-1]$ ya $P(j)$ realizó $a(j)$.

Propiedad $BARRERA$ a garantizar:

$$b(i) \text{ se ejecuta después de } \textbf{todos} \text{ los } a(j)$$

Es decir, queremos *poner una barrera* entre los a y los b. Pero esto debe hacerse sin restringir de más: no hay que imponer ningún orden entre los $a(i)$ ni entre los $b(i)$.

Solución de Barrera con estados RTCE

```
1  atomic<int> cant = 0;   // Procs que terminaron a
2  semaphore barrera = 0; // Barrera baja
3
4  proc P(i) {
5    // R:
6    a(i);
7    // A: terminó a
8    // T:
9    if (cant.getAndInc() < N-1)
10     barrera.wait();
11   // C: no hay nada que hacer
12   // E:
13   barrera.signal();
14   // B: se puede hacer b
15   b(i);
16 }
```

Demostraciones

Ahora demostremos que vale la propiedad *BARRERA* y *G-PROG*.

¡Al pizarrón!