Modelli di interazione tra processi

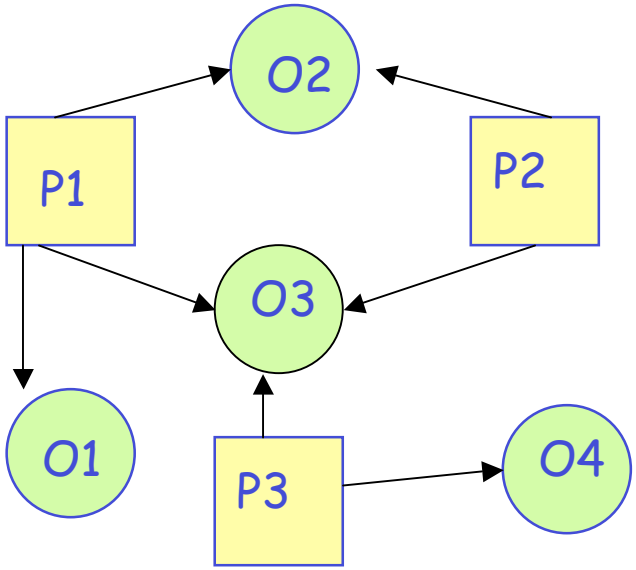
- Modello a memoria comune (ambiente globale, global environment)

- Modello a scambio di messaggi (ambiente locale, message passing)

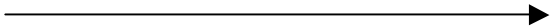
Modello a memoria comune

Il sistema è visto come un insieme di

- **processi**
- **oggetti** (*risorse*)



Diritto di accesso



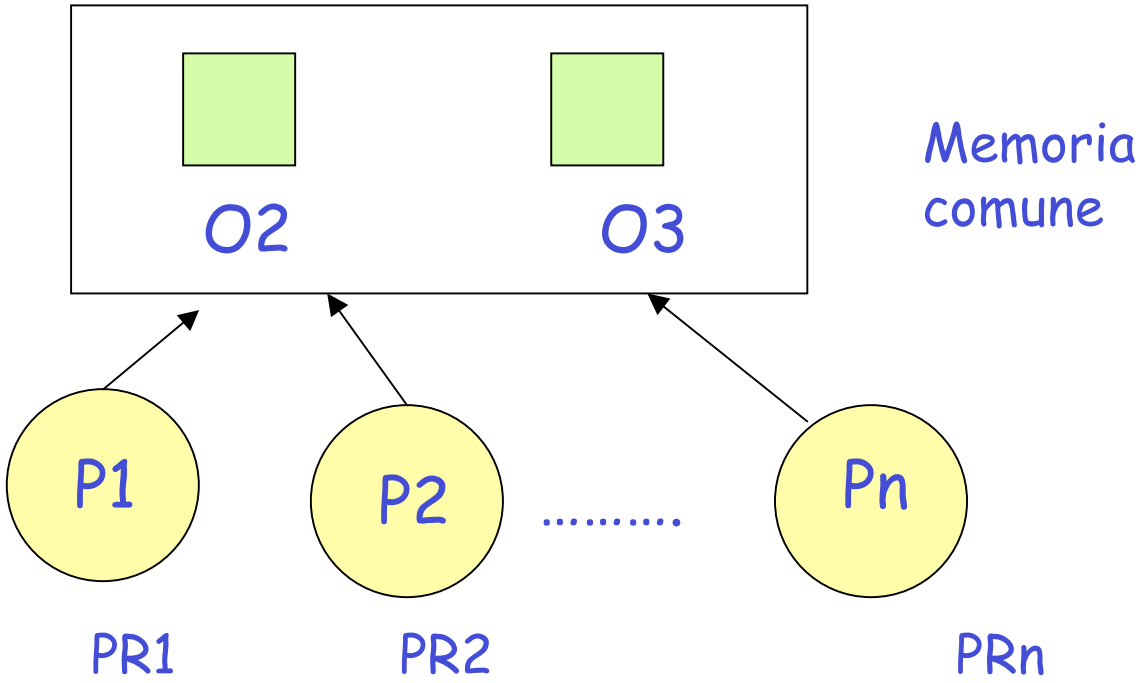
O1 , O4 risorse private

O2 , O3 risorse comuni

Tipi di interazioni tra processi:

- **competizione**
- **cooperazione**

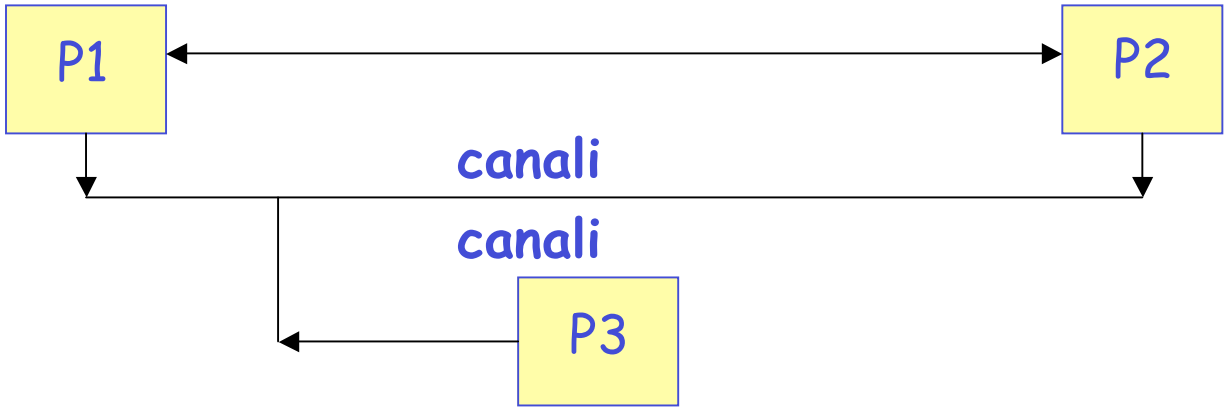
- Il modello a memoria comune rappresenta la naturale astrazione del funzionamento di **un sistema in multiprogrammazione** costituito da uno o più processori che hanno accesso ad una memoria comune



- Ad ogni processore può essere associata una memoria privata, ma ***ogni interazione avviene tramite oggetti contenuti nella memoria comune.***

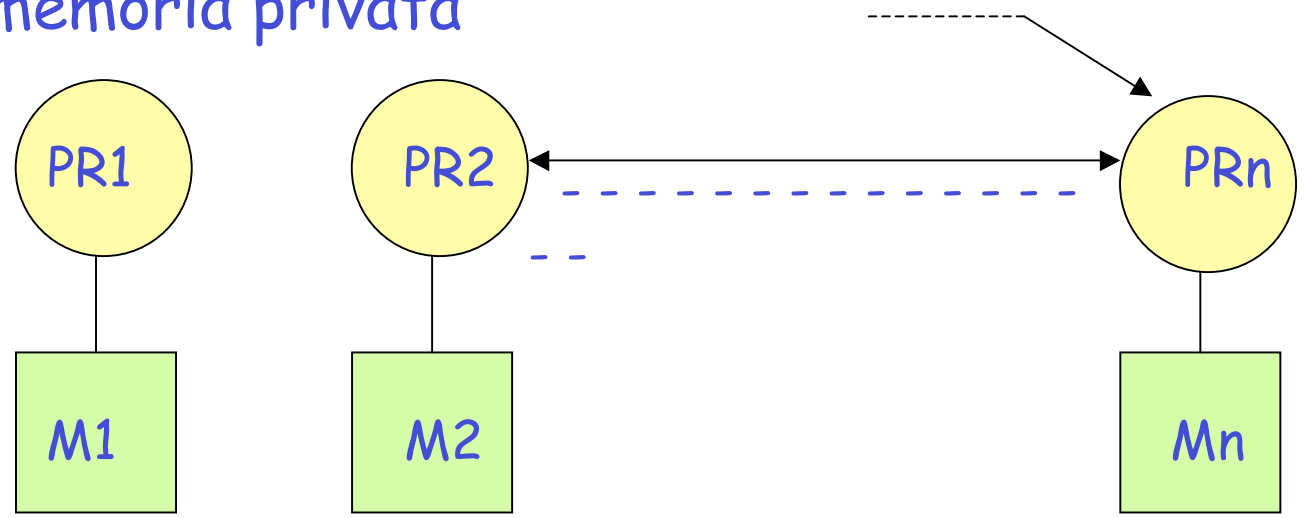
Modello a scambio di messaggi

Il sistema è visto come un insieme di processi ciascuno operante in un *ambiente locale* che non è accessibile a nessun altro processo.



- Ogni forma di interazione tra processi avviene tramite scambio di messaggi.
- Non esiste più il concetto di **risorsa accessibile direttamente ai processi**; sono possibili due casi:
  1) alla risorsa è associato un **processo servitore**
  2) la risorsa viene passata da un processo all'altro **sotto forma di messaggi**

- Il modello a scambio di messaggi rappresenta la naturale astrazione di un sistema *privo di memoria comune*, in cui a ciascun processore è associata una memoria privata



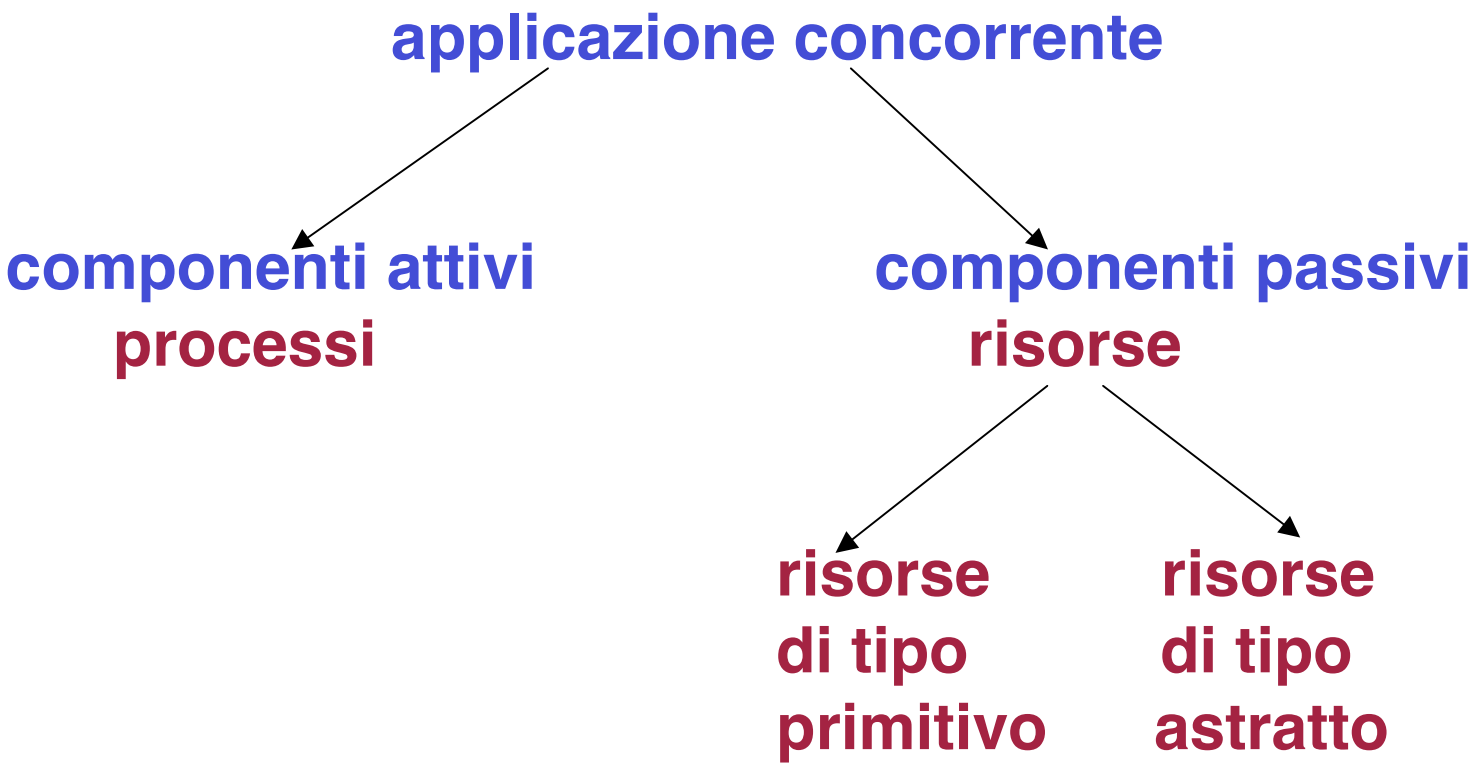
- Il modello a scambio di messaggi può essere realizzato *anche in presenza di memoria comune*, che viene utilizzata per realizzare canali di comunicazione.

Modello a memoria comune

Aspetti caratterizzanti

Ogni applicazione viene strutturata come un insieme di componenti, suddiviso in due sottoinsiemi disgiunti:

- **processi** (componenti attivi)
- **risorse** (componenti passivi).



Risorsa: qualunque oggetto, fisico o logico, di cui un processo necessita per portare a termine il suo compito.

- Le risorse sono raggruppate in *classi*; una classe identifica l'insieme di *tutte e sole* le operazioni che un processo può eseguire per operare su risorse di quella classe.

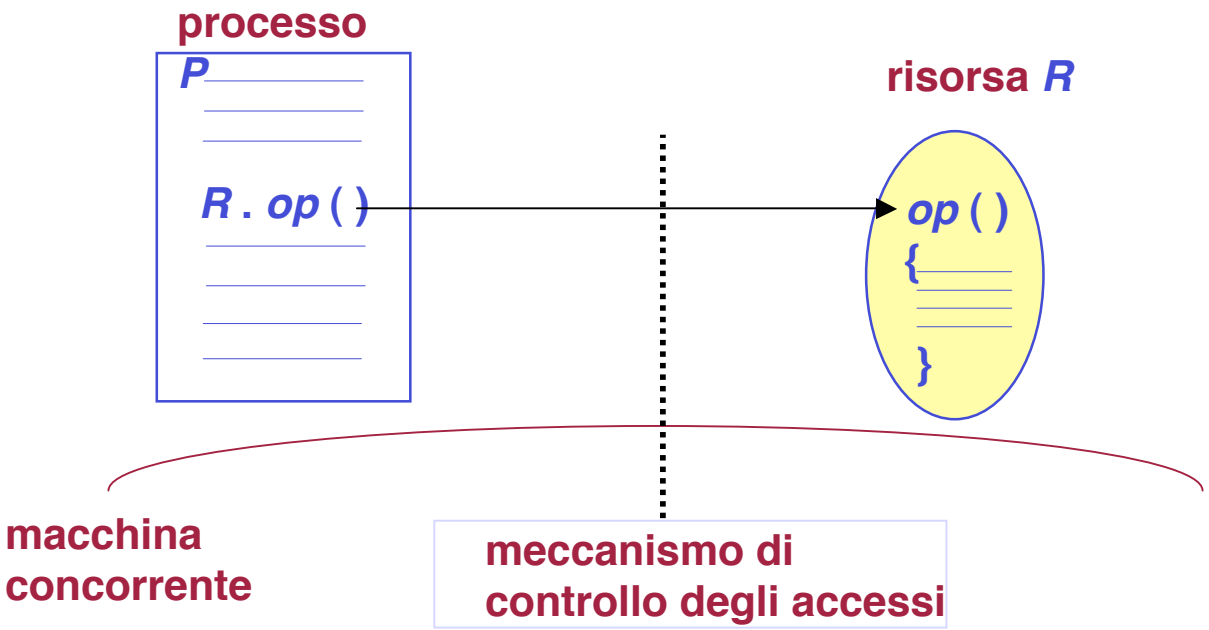
- Il termine risorsa si identifica con quello di *struttura dati* allocata nella memoria comune.

- Vale anche per risorse fisiche: descrittore del dispositivo

Meccanismo di controllo degli accessi

Necessita` di specificare quali processi ed in quali istanti possono accedere alla risorsa.

Il meccanismo di controllo degli accessi ha il compito di controllare che gli accessi dei processi alle risorse avvengano *correttamente*.



Gestore di una risorsa

Per ogni risorsa R, il suo gestore definisce, in ogni istante t, l'insieme *SR(t)* dei processi che, in tale istante, hanno il diritto di operare su R.

Classificazione risorse:

- Risorsa R **dedicata**: se SR (t) ha una cardinalità sempre ≤ 1
- Risorsa R **condivisa**: in caso contrario

- Risorsa R **allocata staticamente**: se SR(t) è una costante:

$$SR(t) = SR(t0), \forall t$$

- Risorsa R **allocata dinamicamente**: se SR (t) è funzione del tempo

Tipologie di allocazione delle risorse

	risorse dedicate	risorse condivise
risorse allocate staticamente	A risorse private	B risorse comuni
risorse allocate dinamicamente	C risorse comuni	D risorse comuni

Per ogni risorsa allocata staticamente, l'insieme SR(t) è definito prima che il programma inizi la propria esecuzione.

Il gestore della risorsa è in questo caso il programmatore che, in base alle regole di visibilità del linguaggio, stabilisce quale processo può "vedere" e quindi operare su R.

Per ogni risorsa R allocata dinamicamente, il relativo gestore G_R definisce l'insieme SR(t) in fase di esecuzione e quindi deve essere un componente della stessa applicazione.

Compiti del gestore di una risorsa

1. mantenere aggiornato l'insieme *SR(t)* e cioè lo stato di allocazione della risorsa;
2. fornire i meccanismi che un processo può utilizzare per acquisire il diritto di operare sulla risorsa, entrando a far parte dell'insieme *SR(t)*, e per rilasciare tale diritto quando non è più necessario;
3. implementare la strategia di allocazione della risorsa e cioè definire quando, a chi e per quanto tempo allocare la risorsa.

Gestore di una risorsa

data una risorsa R,
il suo gestore G_R è costituito da:

una risorsa condivisa
in un sistema
organizzato secondo
il modello a
memoria comune

un processo
in un sistema
organizzato secondo
il modello a
scambio di messaggi

Pool di risorse equivalenti

Sono insiemi di istanze di risorse dello stesso tipo e quindi in grado di svolgere le stesse funzioni.

Vengono gestite da un solo gestore.

Consideriamo un processo P che deve operare, ad un certo istante, su R di tipo T (op1,op2,..,opn):

- Se R è allocata **staticamente** a P (modalità A e B) , il processo possiede il diritto di operare in qualunque istante:

  R.op$_i$(...); /*esecuzione dell'operazione op$_i$ su R*/

- Se R è allocata **dinamicamente** a P (modalità C e D), è necessario prevedere il gestore *GR*, che implementa le funzioni di *Richiesta* e *Rilascio* della risorsa; il processo P deve eseguire il seguente protocollo:

  GR.Richiesta (..); /* acquisizione della risorsa R*/
  R.opi(..); /*esecuzione dell'operazione op$_i$ su R*/
  GR.Rilascio(...); /* rilascio della risorsa R*/

- Se R è allocata come **risorsa condivisa**, (modalità B e D) è necessario assicurare che gli accessi avvengano in modo non divisibile:
  - ➔Le funzioni di accesso alla risorsa devono essere programmate come una *classe di sezioni critiche* (utilizzando i meccanismi di sincronizzazione offerti dal linguaggio di programmazione e supportati dalla macchina concorrente).

- Se R è allocata come **risorsa dedicata**, (modalità A e C), essendo P l'unico processo che accede alla risorsa, non è necessario prevedere alcuna forma di sincronizzazione.

Tipi di interazione: competizione

- Risorse **condivise** e **allocate staticamente** (modalità B). La competizione tra processi avviene al momento dell'accesso alla risorsa. L'accesso esclusivo è garantito dal meccanismo di mutua esclusione utilizzato nel programmare le funzioni di accesso.

- Risorse **dedicate** e **allocate dinamicamente** (modalità C). La competizione tra processi avviene al momento dell'accesso alle operazioni del gestore (accesso esclusivo alle operazioni del gestore).

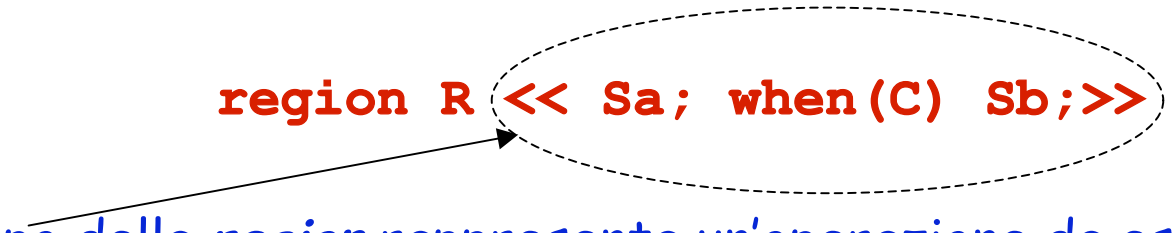
Tipi di interazione: cooperazione

- Risorse **condivise** e **allocate staticamente** (modalità B). Si ha cooperazione se uno dei processi memorizza in R informazioni che possono essere successivamente lette da un altro processo.

- Risorse **dedicate** e **allocate dinamicamente** (modalità C). Si ha cooperazione se un processo, acquisito dal gestore il diritto di operare su R, vi memorizza informazioni e se successivamente il diritto di accesso viene acquisito da un altro processo che legge le informazioni.

Specifica della sincronizzazione

Regione critica condizionale:



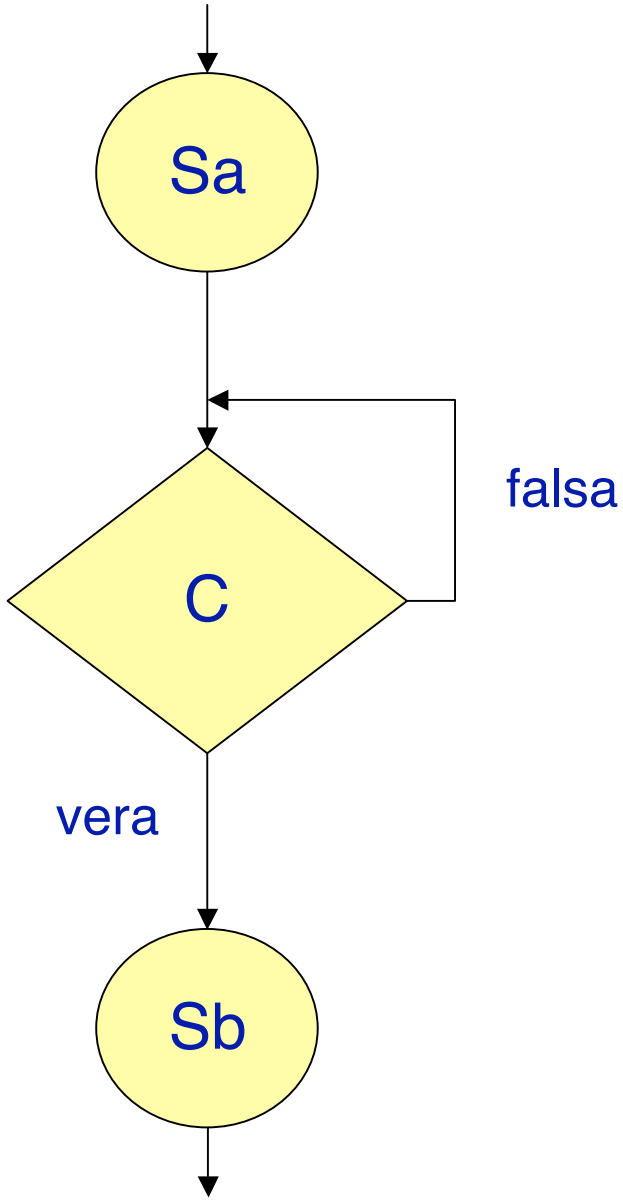
a) il corpo della ***region*** rappresenta un'operazione da eseguire sulla risorsa condivisa **R** e quindi costituisce una **sezione critica** che deve essere eseguita in **mutua esclusione** con le altre operazioni definite su R .

b) il corpo della ***region*** è costituito da due istruzioni da eseguire in sequenza: l'istruzione Sa e, successivamente, l'istruzione Sb.

In particolare,una volta terminata l'esecuzione di Sa viene valutata la condizione *C* :

- se *C* è **vera** l'esecuzione continua con Sb,
- se *C* è **falsa** il processo che ha invocato l'operazione ***attende*** che la condizione *C* diventi vera. A quel punto l'esecuzione della region può riprendere e essere completata mediante l'esecuzione di Sb.

Regione critica



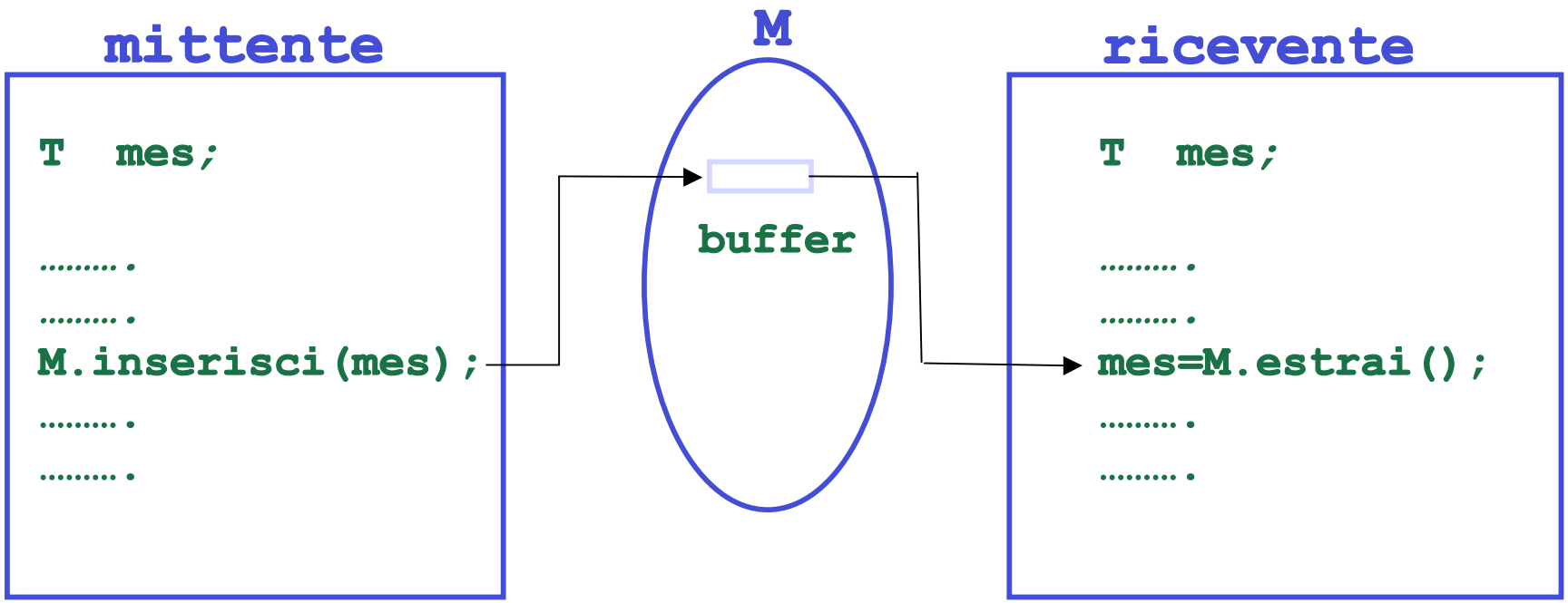
Regioni critiche: casi particolari

1) **region R << S; >>** Specifica della sola **mutua esclusione** senza che sia prevista alcuna forma di sincronizzazione diretta.

2) **region R << when(C) >>** Specifica di un semplice vincolo di sincronizzazione. Il processo deve attendere che C sia verificata prima di proseguire

3) **region R << when(C) S; >>** Specifica il caso in cui la condizione C di sincronizzazione caratterizza lo stato in cui la risorsa R deve trovarsi al fine di poter eseguire l'operazione S.

Scambio di informazioni tra processi



Scambio di informazioni tra processi: specifica della sincronizzazione

Supponendo che la risorsa M sia una struttura con i seguenti campi:

```
T  buffer;
boolean  pieno;
```

si ha:

```
void inserisci (T dato):
region M << when(pieno==false)
                    buffer=dato;
                    pieno=true;>>

T estrai():
region M << when(pieno==true)
                    pieno=false;
                    return buffer;>>
```

Tecniche di allocazione delle risorse

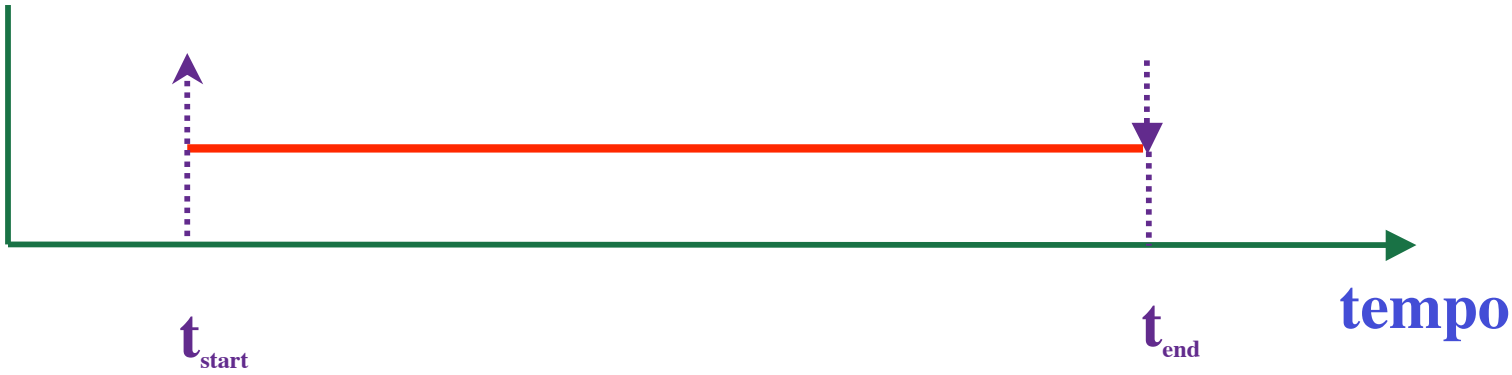
- **Allocazione statica:**

  Il processo mantiene il diritto di operare sulla risorsa (linea continua) dall'istante in cui è creato (tstart) all'istante in cui termina (tend).

- **Allocazione dinamica:**

  Il processo ha il diritto di operare sulla risorsa solamente negli intervalli nei quali gli e` allocata la risorsa.

Allocazione statica



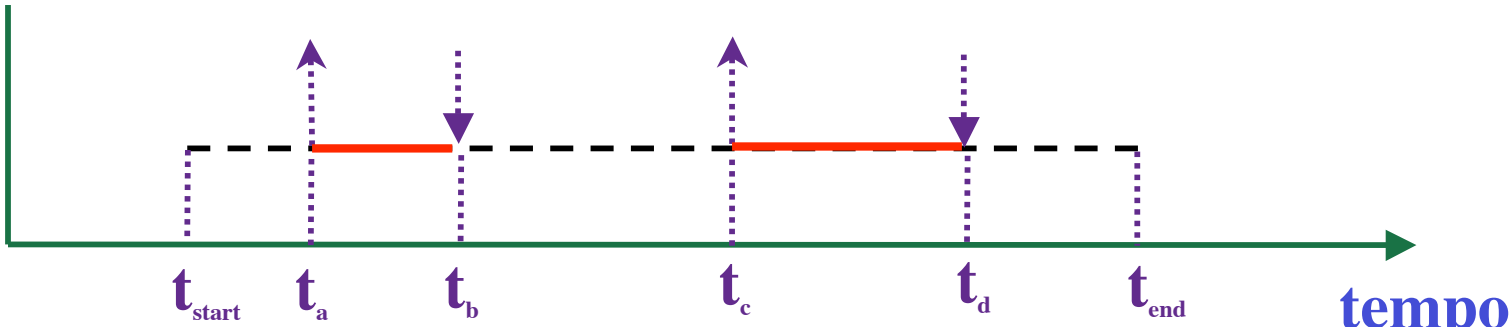
allocazione

deallocazione

t_{start} creazione processo

t_{end} terminazione processo

Allocazione dinamica



↑ allocazione

↓ deallocazione

t_{start} creazione processo

t_{end} terminazione processo

Allocazione assoluta:

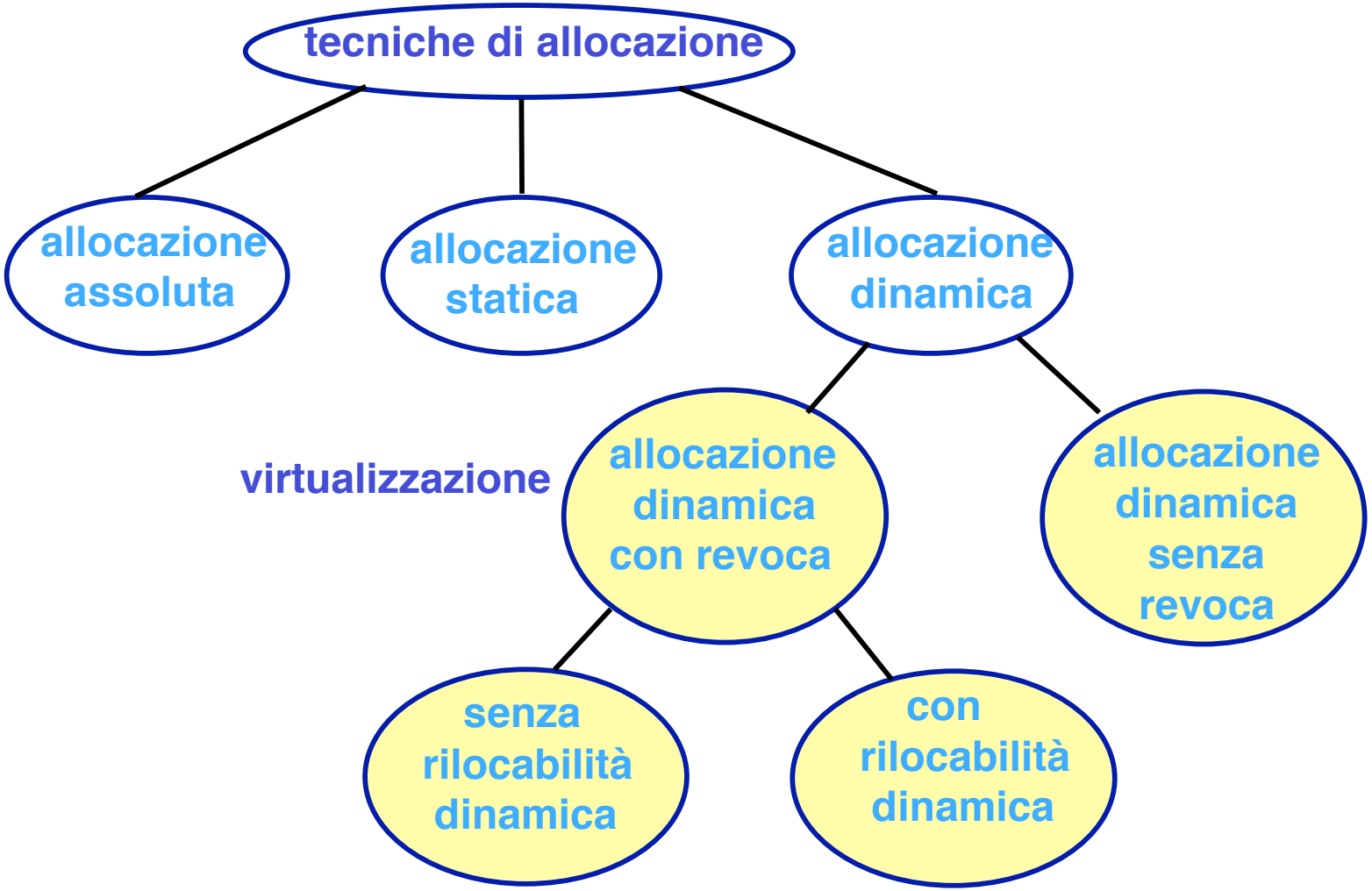
Risorse allocate a processi di sistema (daemon) che non terminano mai , creati cioè in fase di inizializzazione del sistema operativo (bootstrap) e restano in vita fino sllo shut down del sistema.

Allocazione dinamica con revoca:

E' il sistema operativo che decide quando allocare la risorsa e quando deallocarla. Due casi:

- allocazione senza rilocabilità dinamica, se in presenza di un pool di risorse equivalenti si è costretti ad allocare la stessa istanza relativa alla prima allocazione;
- allocazione con rilocabilità dinamica, nel caso contrario.

Tecniche di allocazione delle risorse



Meccanismi linguistici per la programmazione di interazioni

Per presentare *alcuni meccanismi linguistici* usati nella programmazione di interazioni tra processi si seguirà il seguente schema:

1. **Presentazione** del meccanismo
2. **Esempi** di uso del meccanismo
3. **Traduzione** del meccanismo linguistico in termini di un *meccanismo primitivo* di sincronizzazione fornito dal supporto a tempo di esecuzione (***nucleo***)

➔ Il meccanismo primitivo di riferimento è quello **semaforico** con le due operazioni **p** e **v** (Dijkstra).