Introduzione alla Concorrenza

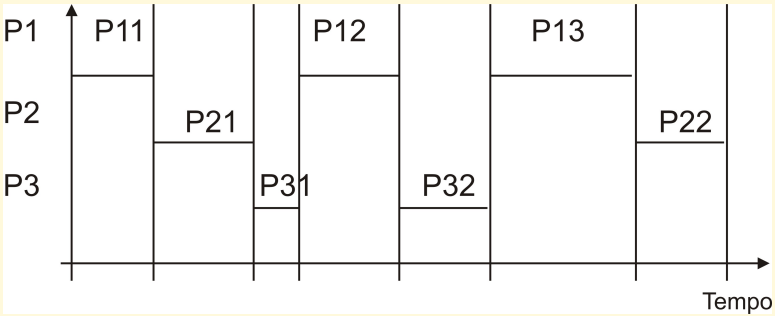
Indice degli argomenti

- generalit, interleaving
- grafi delle precedenze: semplificazione, tempi di calcolo
- strumenti linguistici(coroutine,fork-join,cobegin-coend)
- sincronizzazione
- determinatezza

Introduzione alla Concorrenza

Generalit

- Interleaving: Nel caso di una sola CPU l'esecuzione procede per intersfogliatura (interleaving) delle istruzioni. Nel caso ci siano piú processi l'esecuzione comprende anche delle sovrapposizioni.



Context switch: realizza la concorrenza, cioé realizza quel meccanismo che consente a piú processi di eseguire simultaneamente.

Introduzione alla Concorrenza

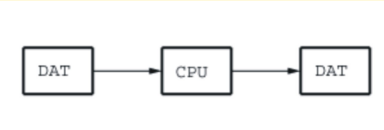
Definizioni

- Atomicitá: esecuzione dall'inizio alla fine senza interferenza
- Contesa (race condition): la situazione nella quale piú processi accedono una risorsa condivisa e il risultato dipende dall'ordine di accesso.
- Attesa indefinita (starvation): la situazione nella quale non c'e' un blocco ma una attesa indefinita.
- Sezione critica : é una sezione di codice che accede ad una risorsa condivisa con un'altro processo.
- Mutua esclusione (mutual exclusion): una risorsa condivisa tra piú processi puó essere usata solo da un processo alla volta. Non ci possono essere piú processi che usano simultaneamente quella risorsa.
- Stallo (deadlock): situazione di blocco di due processi nella quale ciascun processo aspetta che l'altro faccia qualcosa prima di eseguire.

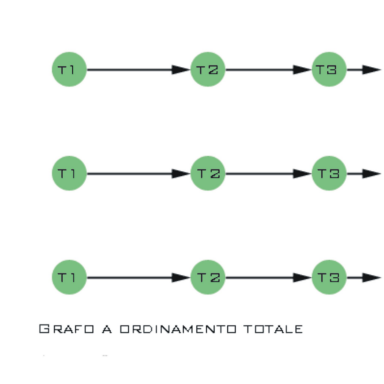
Introduzione alla Concorrenza

Grafi delle precedenze: costruzione, semplificazione, tempi di calcolo, determinatezza

Elaborazione sequenziale di tre processi: uno legge, uno elabora e uno scrive da un nastro



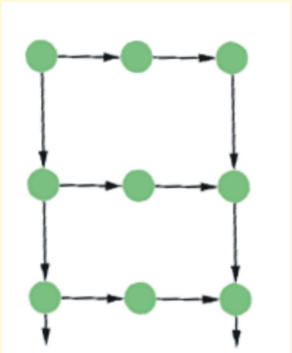
Elaborazione seguenziale di tre elaborazioni

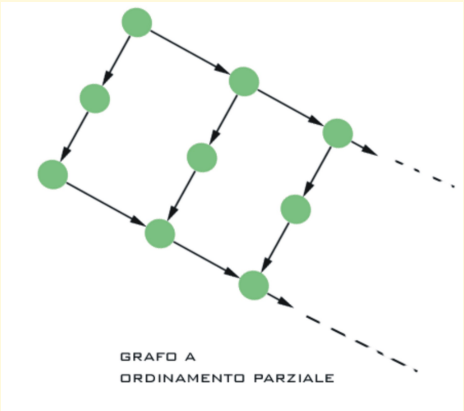


Introduzione alla Concorrenza

Grafi delle precedenze: costruzione, semplificazione, tempi di calcolo, determinatezza

Elaborazione concorrente di tre elaborazioni





Introduzione alla Concorrenza

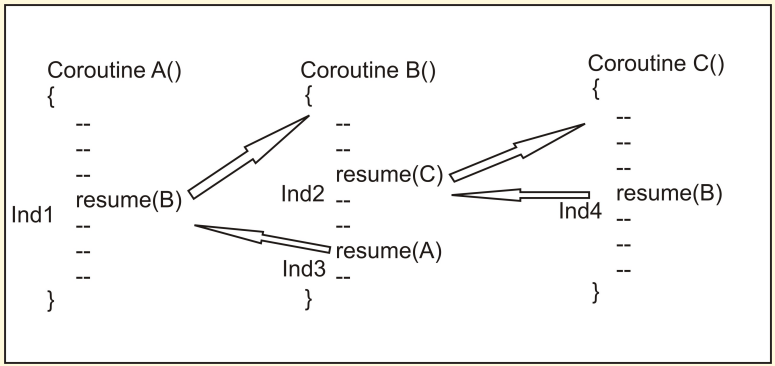
Interazioni tra threads

- **A)** COOPERAZIONE: i singoli task si possono passare delle informazioni
- **B)** CONTESA: la stessa risorsa pu essere richiesta in contemporanea da due task, rendendo cos necessaria la presenza di un **arbitro**
- **C)** INTERFERENZA: task che interferiscono in maniera non desiderata dal programmatore [a]
- **D)** SINCRONIZZAZIONE. Ad un certo istante, piú processi concorrenti devono attendere l'un l'altro in modo tale da realizzare una determinata operazione. La sincronizzazione non é legata solo alla protezione di una risorsa condivisa, ma anche al rispetto di un ordine di funzionamento.

[a] sono errori di programmazione

Introduzione alla Concorrenza

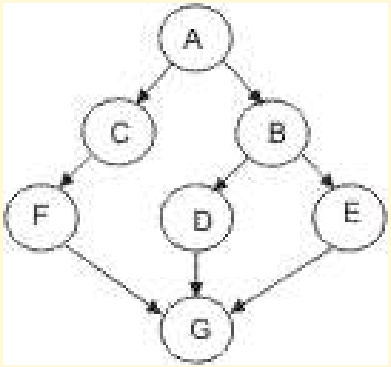
Strumenti linguistici(coroutine)



Introduzione alla Concorrenza

Strumenti linguistici(cobegin-coend)

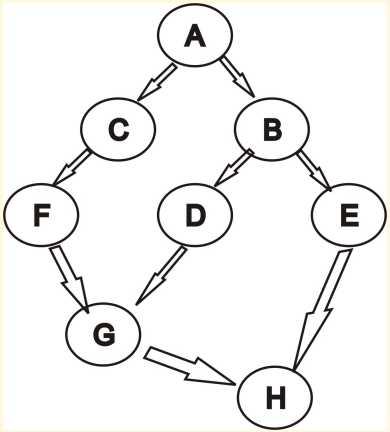
```
cobegin
    P1(); P2(); P3();
coend;
```




```
begin
   A;
   cobegin
      begin A; F; end;
      begin
            B;
            cobegin
               D; E;
            coend;
      end;
    coend;
    G;
end;
```

Strumenti linguistici(fork-join)

Alcuni grafi non possono essere realizzati con cobegin-coend -¿ fork-join:



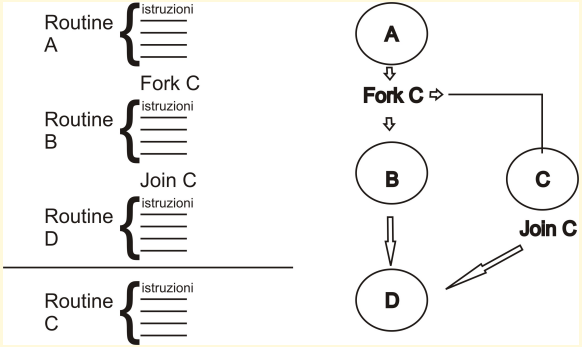

```
processo P;

P = fork(routine);

join P;
```

Introduzione alla Concorrenza

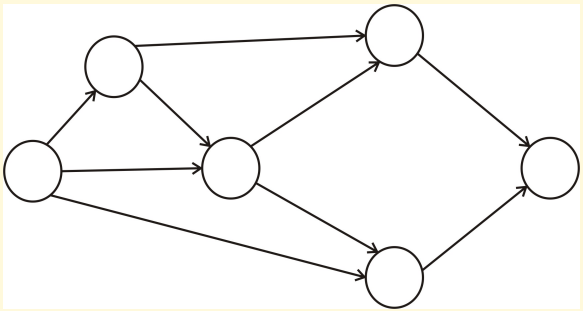
Meccanismo di fork-jon

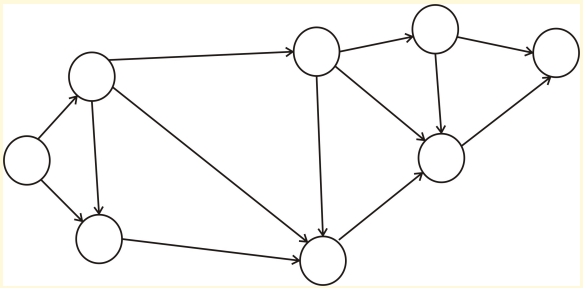


Introduzione alla Concorrenza

Semplificazione: eliminazione di precedenze implicite

Esempi di grafi da semplificare

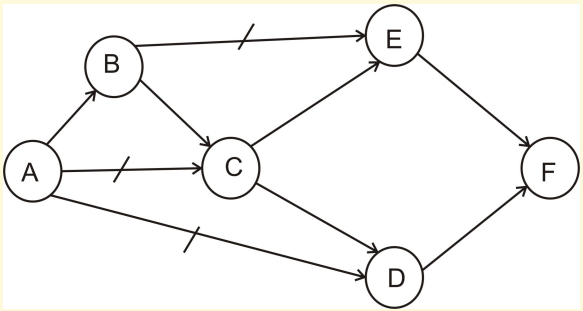


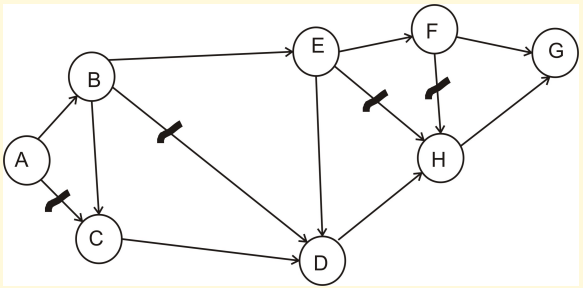


Introduzione alla Concorrenza

Semplificazione: risultato

Esempi di grafi da semplificare

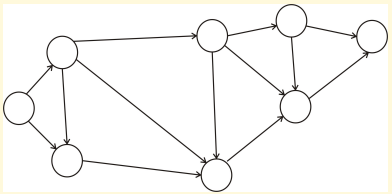


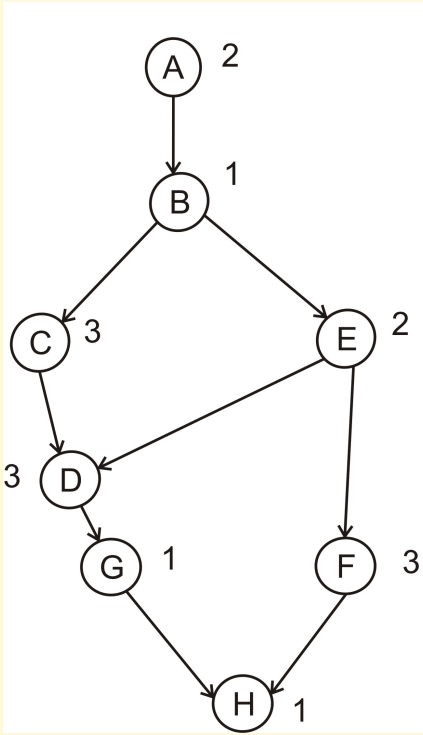


Introduzione alla Concorrenza

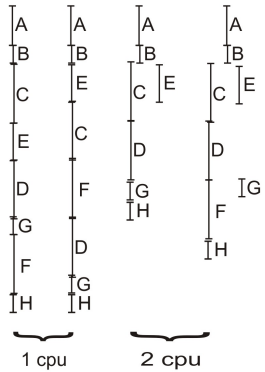
Tempo di calcolo

dipende dai vincoli e dai tempi di calcolo dei singoli task e dal grado di parallelismo





Diverse sequenze di esecuzione



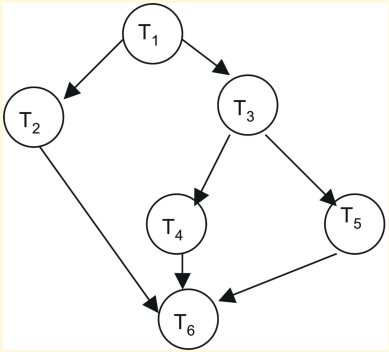
Determinatezza: alcune definizioni

- insieme di thread: $\Gamma = \{T_1, \ldots, T_n\}$
- eventi fondamentali associati ad un thread: inizio $\overline{T}$ e terminazione $\underline{T}$. $t(\underline{T}) - t(\overline{T}) > 0$ dove $t(.)$ é il tempo del thread
- relazione d'ordine parziale su Γ: $T \prec T'$
- sistema di thread: $C = (\Gamma, \prec)$
- grafo di precedenza: (Γ, archi orientati che rappresentano i vincoli). L'arco (T, T') da T a T' $\in$ grafo se e solo se T $\prec$ T' e $\nexists$ T" : T $\prec$ T" $\prec$ T'
- nodi predecessori, successori;
- nodi immediati predecessori, immediati successori; nodi terminali, iniziali
- nodi indipendenti. T e T' possono essere concorrenti se e solo se sono indipendenti.
- **Sistema di thread chiuso**
- **Concatenazione, Combinazione parallela** di sistemi di thread

Determinatezza: definizioni (cont.)

- sequenza di esecuzione α di un sistema di n thread
- Per ogni T in Γ i simboli $\underline{T}$ e $\overline{T}$ appaiono esattamente una volta in α.
- Se $a_i = \overline{T}$ e $a_j = \underline{T}$ allora $i < j$.
- Se $a_i = \underline{T}$ e $a_j = \overline{T}$, dove T $\prec$ T' allora $i < j$.

Esempio: sequenze di esecuzione valide per



puó essere

$$\overline{T_1}\underline{T_1}\overline{T_2}\underline{T_2}\overline{T_3}\underline{T_3}\overline{T_4}\underline{T_4}\overline{T_5}\underline{T_5}\overline{T_6}\underline{T_6}$$

ma anche

$$\overline{T_1}\underline{T_1}\overline{T_2}\overline{T_3}\underline{T_3}\overline{T_4}T_5\underline{T_2}T_4\underline{T_5}\overline{T_6}\underline{T_6}$$

Determinatezza di un sistema di thread

Definizione 1 *Per convenienza, chiamamo* $V(M_i, \alpha)$ *la sequenza di valori scritti nella cella* M_i *dalla sequenza di esecuzione* α.

Definizione 2 *Un sistema di thread* **C** *é* **determinato** *se per ogni* s_0, $V(M_i, \alpha) = V(M_i, \alpha')\forall i = 1 \ldots m$ *e per tutte le sequenze di esecuzione.*

Definizione 3 *Due thread T e T' sono* **noninterferenti** *se T un successore o predecessore di T' oppure se* $R_T \cap R_{T'} = R_T \cap D_{T'} = D_T \cap R_{T'} = 0$.

Definizione 4 *Un insieme di Thread é* **mutualmente non interferente** *se Ti e Tj sono nonintereferenti per tutti gli indici i e j* $(i \neq j)$.

Teorema 1 *Condizione sufficiente per la determinatezza di un sistema di n thread che il sistema di thread sia mutualmente non interferente.*

Esempio di Indeterminatezza di una applicazione multithread

Thread T1

```
void echo ( )
{
in = getchar( );
out = in;
cout << out;
}
```

Thread T2

```
void proc ( )
{
out = in + "...";
cout << out;
}
```

Thread T3

```
void stam ( )
{
out = in + "---";
cout << out;
}
```

Eseguendo $\overline{T_1}\underline{T_1}\overline{T_2}\underline{T_2}\overline{T_3}\underline{T_3}$ e scrivendo per esempio ”abc” si ha in uscita la scritta: ”abcabc...abc—” ma eseguendo
$\overline{T_1}\underline{T_1}\overline{T_2T_3}\underline{T_3T_2}$
si ha ”abcabc...abc—” e se $\overline{T_1}\underline{T_1}\overline{T_3T_2}\underline{T_3T_2}$
si ha ”abcabc—abc...” cioé una esecuzione non determinata.