

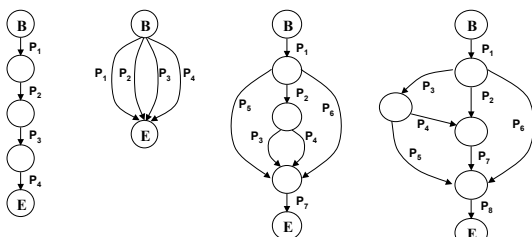
Programowanie współbieżne

Grafy przepływu procesów

Grafy przepływu procesów przedstawiają zależności czasowe wykonywania procesów. Wierzchołki tych grafów reprezentują chwile czasu, natomiast krawędzie zorientowane - procesy. Dwa wierzchołki są połączone krawędzią zorientowaną (łukiem) jeżeli istnieje proces, którego moment rozpoczęcia odpowiada pierwszemu wierzchołkowi, a moment zakończenia - drugiemu.



Przykład grafów



Legenda:

B-begin
E-end

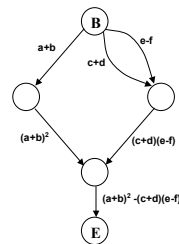
Dobrze zagnieżdżony graf przepływu procesów

Graf przepływu procesów jest **dobrze zagnieżdżony**, jeżeli może być opisany przez funkcje $P(a,b)$ i $S(a,b)$ lub ich złożenie, gdzie $P(a,b)$ i $S(a,b)$ oznaczają odpowiednio wykonanie równoległe i szeregowe procesów a i b .

Przykład

Przedstawić graf przepływu procesów odpowiadających wyznaczeniu wyrażenia

$$y := (a + b)^2 - (c + d)(e - f)$$



Notacja "and" (Wirth).

Współbieżne wykonanie może być specyfikowane za pomocą operatora **and**, który łączy dwa wyrażenia wykonywane współbieżnie.

Przykład:

```
...
begin
  x1 := x1 + 2;
  y1 := x1 + y1;
end
and
  x2 := 2 * x2 + y2;
y2 := x1 + x2 + y1 + y2;
...
```

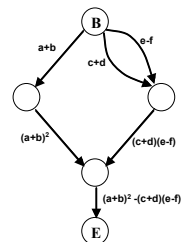


Przykład notacji „and”

Zastosowanie notacji and do implementacji programu wyznaczającego wartość wyrażenia:

$$(a+b)^2 - (c+d)(e-f)$$

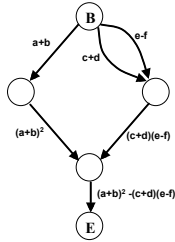
```
begin
  begin
    x1 := a + b;
    x2 := x1 * x1;
  end
  and
    begin
      x3 := c + d
      and
        x4 := e - f;
      x5 := x3 * x4;
    end;
  x6 := x2 - x5;
end.
```



Notacja "parbegin - parend" ("cobegin - coend", Dijkstra)

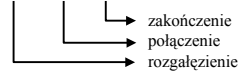
Wszystkie wyrażenia ujęte w nawiasy parbegin – parend są wykonywane wspólnie

```
begin
  parbegin
    begin
      x1:=a+b;
      x2:=x1*x1;
    end;
    begin
      parbegin
        x3:=c+d;
        x4:=e-f;
      parend;
      x5:=x3*x4;
    end;
  parend;
  x6:=x2-x5;
end.
```



Notacja "fork, join, quit" (Conway)

fork join quit



- ❑ Instrukcja **quit** powoduje zakończenie procesu.
- ❑ Instrukcja **fork w** oznacza, że proces w którym wystąpiła ta instrukcja będzie dalej wykonywany wspólnie z procesem identyfikowanym przez etykietę w.
- ❑ Instrukcja **join t, w** ma dwa argumenty, z których *t* jest licznikiem a *w* - etykietą

Wykonanie instrukcji *join t, w* oznacza:

$t := t - 1;$

if $t = 0$ **then go to** *w*;

przy czym sekwencja tych dwóch instrukcji jest wykonywana atomowo, tzn. że jest niepodzielna (instrukcja jest wykonana w całości albo wcale).

Przykład notacji „fork, join, quit”

```
begin
  t1:=2;      w1:  x1:= a + b;
  t2:=2;      x2:= x1 * x1;
  fork w1;    join t1, w5;
  fork w2;    quit;
  fork w3;    w2:  x3:= c + d;
  quit;       join t2, w4;
  w3:  x4:= e - f;
  join t2, w4;
  quit;
  w4:  x5:= x3 * x4;
  join t1, w5;
  quit;
  w5:  x6:= x2 - x5;
  quit;
end.
```

