

Wykład 2

Podstawowe pojęcia związane z przetwarzaniem równoległym, Zależności w programach, 2 godz.

Plan

1. Co to jest zależność
2. Rodzaje zależności w pętlach
3. Tożsamość semantyczna programów
4. Wpływ zależności na tożsamość semantyczną programów

Zależności

Zależności istniejące w programie dzielimy na zależności danych i zależności sterowania.

Zależności

Zależność danych jest sytuacją, w której wynik wykonywania jakiejś instrukcji programu jest uzależniony od wyniku poprzedniej instrukcji (jakaś instrukcja programu odwołuje się do wyniku poprzedniej instrukcji).

Zależności

Takie dwie instrukcje nazywamy zależnymi lub mówimy że jedna instrukcja zależy od drugiej.

Zależności

Instrukcja S_2 jest zależna od instrukcji S_1 jeśli jest spełniony warunek Bernstein'a:

$$[I(S_1) \cap O(S_2)] \cup [O(S_1) \cap I(S_2)] \cup [O(S_1) \cap O(S_2)] \neq \emptyset$$

suma zbiorów

przecięcie zbiorów

gdzie:

Input

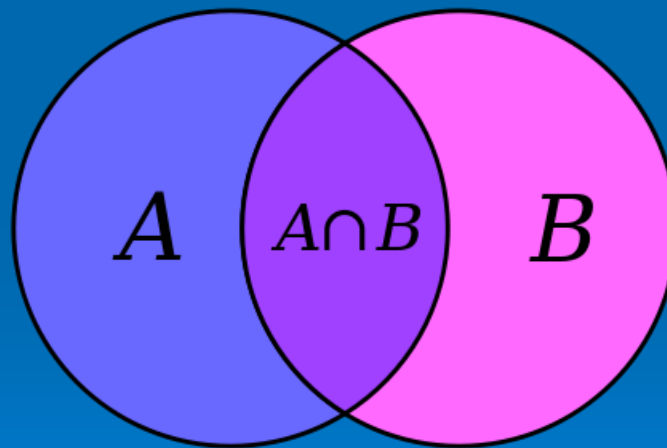
Output

$I(S_i)$ jest zbiorem komórek pamięci odczytywanych przez instrukcję S_i ,

$O(S_j)$ jest zbiorem komórek pamięci, do których S_j zapisuje wynik, i istnieje ścieżka obliczeń z S_1 do S_2 .

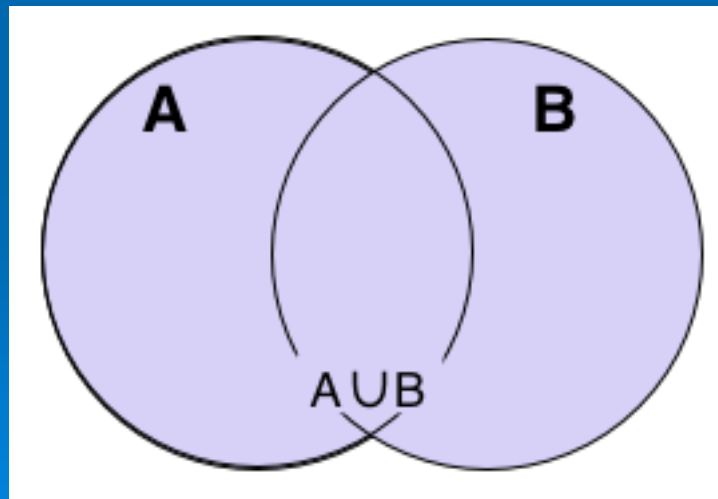
Zależności

\cap oznacza przecięcie zbiorów – zbiór, który zawiera te i tylko te elementy, które należą jednocześnie do obydwu zbiorów



Zależności

U oznacza sumę zbiorów - zbiór złożony ze wszystkich elementów należących do któregośkolwiek z sumowanych zbiorów.



Zależności

Możemy wyróżnić trzy przypadki zależności:

1. Zależność przepływu danych:

$$O(S_1) \cap I(S_2) \neq \emptyset$$

I(Input) kojarzymy z odczytywaniem danych
O(Output) kojarzymy z zapisywaniem danych (ostatnia operacja przed wyjściem)

$S_1 \rightarrow S_2$: S_1 zapisuje wynik odczytywany przez S_2

Zależności

Możemy wyróżnić trzy przypadki zależności:

2. Zależność odwrotna:

$$I(S_1) \cap O(S_2) \neq \emptyset$$

$S_1 \rightarrow S_2$: S_1 czyta dane wcześniej niż S_2
nadpisze je

Zależności

Możemy wyróżnić trzy przypadki zależności:

3. Zależność po wyjściu:

$$O(S_1) \cap O(S_2) \neq \emptyset$$

$S_1 \rightarrow S_2$: S_1 i S_2 zapisują do tej samej komórki pamięci

Zależności

Zależność sterowania

Zależność sterowania ma miejsce wtedy, gdy możliwość wykonywania jakiejś instrukcji jest uzależniona od wyniku poprzedniej instrukcji lub operatora.

Zależności

Przykład zależności sterowania:

S_1 : if (a == b)

S_2 : a = a + b;

S_3 : b = a + b;

Możliwość wykonywania S_2 jest uzależniona od wyniku operacji (a == b).

Zależności

Zmienne skalarne

1. Zależności przepływu danych lub zapis- odczyt (*Data flow dependences*)

a=b (1)
↓
c=a (2)

a=b (1)
↓
.....
↓
c=a (2)

Może być dowolna
sekwencja
instrukcji między
(1) a (2)

Zależności

Zmienne skalarne

2. Zależności odwrotne lub odczyt – zapis (*Anti dependences*)

a=b (1)
↓
b=c (2)

a=b (1)
↓
.....
b=c (2)

Zależności

Zmienne skalarne

3. Zależności po wyjściu lub zapis-zapis (*Output dependences*)

a=b (1)
↓
a=c (2)

a=b (1)
↓
a=c (2)

Zależności

Zmienne skalarne

4. Zależności po wejściu lub odczyt-odczyt(*Input dependences*)

a=b (1)



c=b (2)

a=b (1)




c=b (2)

Zależności

Zmienne tablicowe

1. Zależności przepływu danych

$$a[1]=b[1] \quad (1)$$

$$c[1]=a[1] \quad (2)$$


Zależności

Zmienne tablicowe

Żeby istniała zależność między instrukcjami, dwa warunki muszą być spełnione:

- 1) ta sama nazwa zmiennej ma miejsce w obydwu instrukcjach (**a** dla naszego przykładu),
- 2) ta sama wartość indeksu ma miejsce w obydwu instrukcjach (**1** dla naszego przykładu).

Zależności

Zmienne tablicowe

2. Zależności odwrotne

$$a[1]=b[1] \quad (1)$$

$$b[1]=c[1] \quad (2)$$




Zależności

Zmienne tablicowe

3. Zależności po wyjściu

$$a[1]=b[1] \quad (1)$$

$$a[1]=c[1] \quad (2)$$


Zależności

Zmienne tablicowe

$$a[i]=b[i] \quad (1)$$

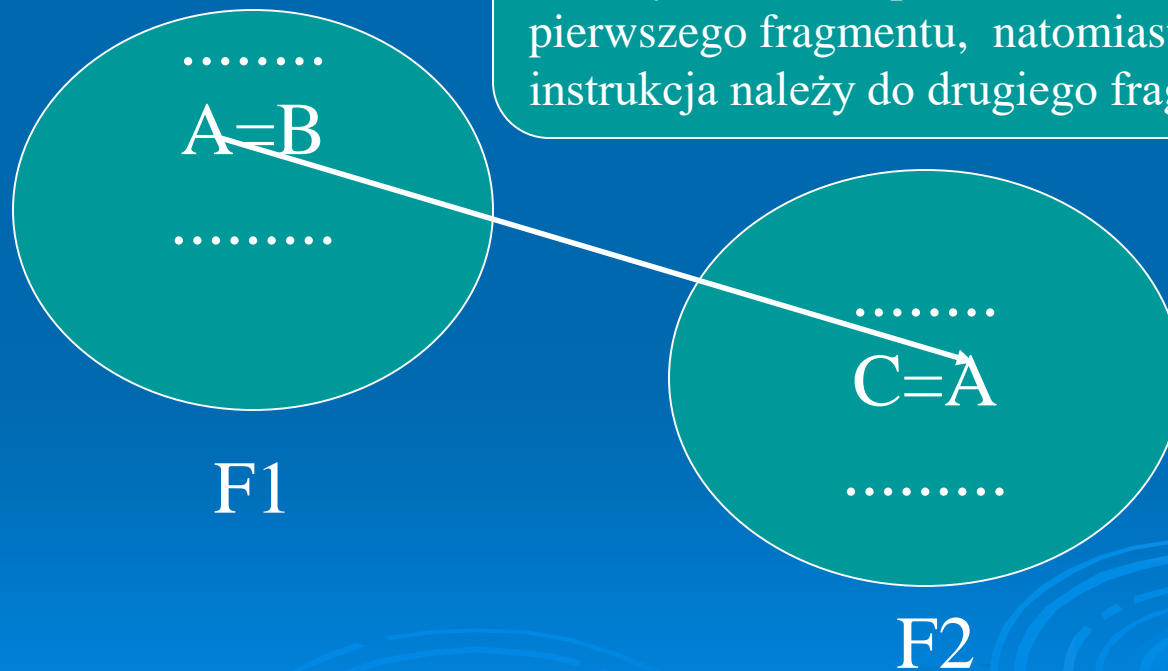
$$c[i]=a[i] \quad (2)$$

Indeksem może być nie tylko stała, ale również nazwa (**i** w naszym przykładzie).

Zależności

Fragmenty kodu

Dwa fragmenty kodu są zależne wtedy, gdy istnieje co najmniej jedna para instrukcji zależnych taka, że pierwsza z nich należy do pierwszego fragmentu, natomiast druga instrukcja należy do drugiego fragmentu.



Zależności

Pętle

```
for(i=0; i<=n; i++)  —————>  nagłówek  
{  
    a[i]=b[i];          —————>  ciało  
}
```

Iteracja jest to pojedyncze wykonanie ciała pętli dla danej wartości indeksu i .

Zależności

Pętle

```
for(i=0; i<=n; i++)  
{  
    a[i]=b[i];  
}
```

a[0]=b[0] pierwsza iteracja

a[1]=b[1] druga iteracja

a[2]=b[2] trzecia iteracja

Zależności

Pętle

Zależności mogą mieć miejsce:

- 1) Pomędzy instrukcjami w ciele pętli,
- 2) Pomędzy iteracjami pętli.

Zależności

Pętle

Przykład 1.

```
for(i=0; i<=n; i++) {  
    a[i]=b[i];  
    c[i]=a[i];}
```

Zależności

Pętle

Przykład 1.

```
for(i=0; i<=n; i++) {  
    a[i]=b[i];  
    c[i]=a[i];  
}
```

a[0] = b[0]

c[0] = a[0]

a[1] = b[1]

c[1] = a[1]



It.1



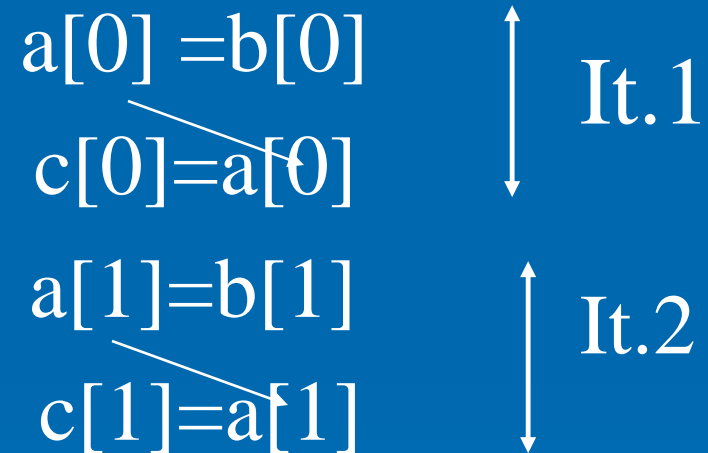
It.2

Zależności

Pętle

Przykład 1.

```
for(i=0; i<=n; i++) {  
    a[i]=b[i];  
    c[i]=a[i];  
}
```



Zależności pomiędzy instrukcjami w ciele pętli.

Brak zależności pomiędzy iteracjami.

Zależności

Pętle

Przykład 2.

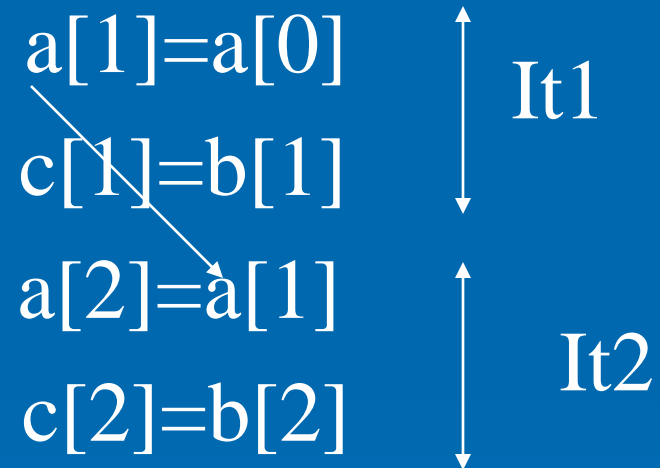
```
for(i=1; i<=n; i++) {  
    a[i]=a[i-1];  
    c[i]=b[i];  
}
```

Zależności

Pętle

Przykład 2.

```
for(i=1; i<=n; i++) {  
    a[i]=a[i-1];  
    c[i]=b[i];  
}
```

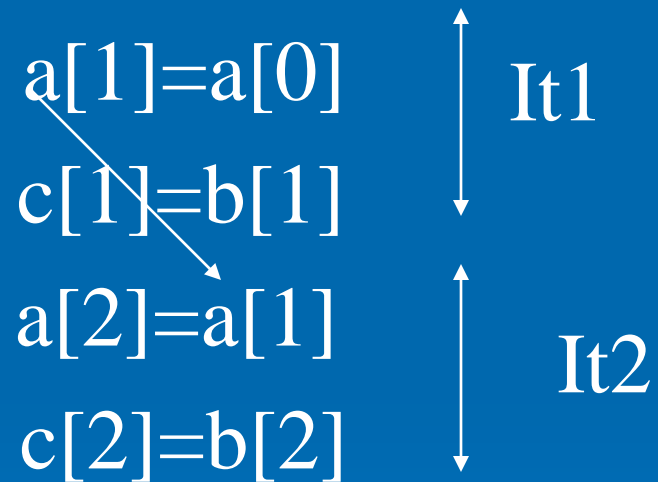


Zależności

Pętle

Przykład 2.

```
for(i=1; i<=n; i++) {  
    a[i]=a[i-1];  
    c[i]=b[i];  
}
```



Brak zależności pomiędzy instrukcjami w ciele pętli.

Występują zależności pomiędzy iteracjami.

Zależności

Pętle

Przykład 3.

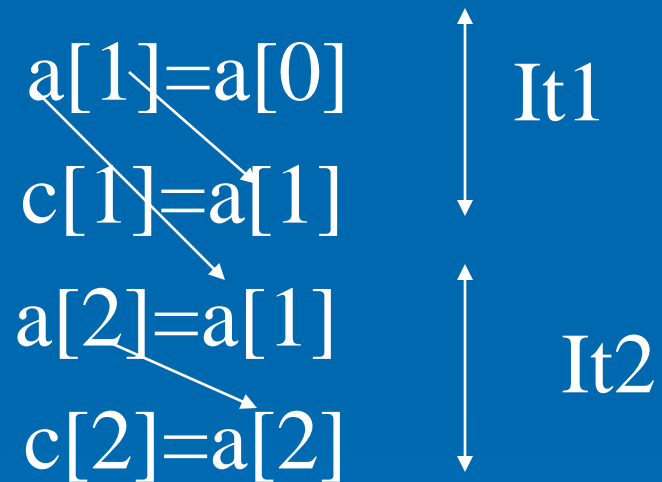
```
for(i=1; i<=n; i++) {  
    a[i]=a[i-1];  
    c[i]=a[i];  
}
```

Zależności

Pętle

Przykład 3.

```
for(i=1; i<=n; i++) {  
    a[i]=a[i-1];  
    c[i]=a[i];  
}
```

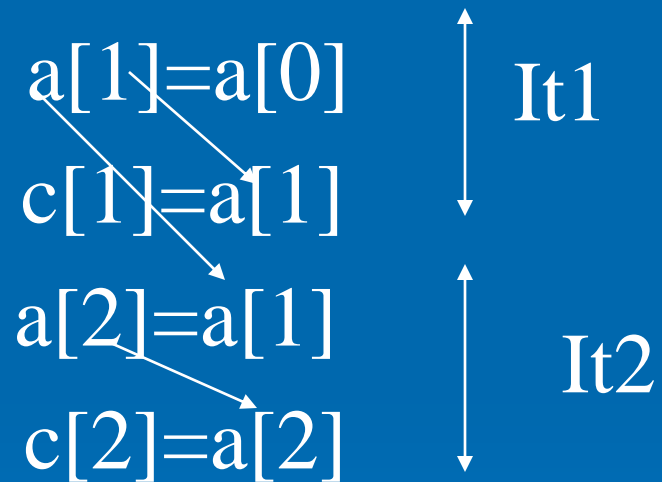


Zależności

Pętle

Przykład 3.

```
for(i=1; i<=n; i++) {  
    a[i]=a[i-1];  
    c[i]=a[i];  
}
```



Zależności pomiędzy instrukcjami w ciele pętli.

Zależności pomiędzy iteracjami.

Zależności

Pętle

Przykład 4.

```
for(i=1; i<=n; i++)  
    for(j=1; j<=n; j++) {  
        a[i][j]=a[i][j-1];  
    }
```

Zależności

Pętle

Przykład 4.

```
for(i=1; i<=n; i++)  
    for(j=1; j<=n; j++) {  
        a[i][j]=a[i][j-1];  
    }
```

a[1][1]=a[1][0] It1

a[1][2]=a[1][1] It2

a[1][3]=a[1][2] It3

.....

a[2][1]=a[2][0]

a[2][2]=a[2][1]

.....

Zależności

Tożsamość semantyczna

Dwa programy są tożsame semantycznie wtedy i tylko wtedy, gdy produkują te same dane wyjściowe dla tych samych danych wejściowych.

Zależności

Tożsamość semantyczna

Jeśli program równoległy jest utworzony w oparciu o program sekwencyjny, to wyniki produkowane przez obydwa programy muszą być takie same.

Zależności

Tożsamość semantyczna

Jeśli istnieje zależność w programie, to zależne instrukcje/iteracje/fragmenty nie mogą być wykonywane równoległe, ponieważ program równoległy może produkować wynik inny niż wynik produkowany przez odpowiedni program sekwencyjny.

Zależności

Tożsamość semantyczna

Tylko niezależne instrukcje/iteracje/fragmenty kodu mogą być wykonywane równolegle !!!

Zależności

Tożsamość semantyczna

Przykład

Przetwarzanie:

a=1; Sekwencyjne

b=2; a=b=2;

c=3; c=a=2;

a=b;
↙
c=a;

Równoległe

a=b=2; c=a=1

Obliczenia sekwencyjne i równoległe produkują różne wyniki dla zależnych instrukcji

Dziękuję za uwagę