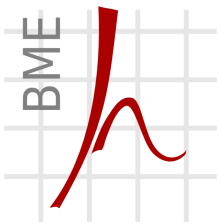
Fordítás – Kódoptimalizálás

Kód visszafejtés.



Híradástechnikai Tanszék

Izsó Tamás

2016. október 13.

Vezérlésfolyam-gráf – Control Flow Gráf (CFG)

- irányított gráf
- csomópontjai az alapblokkok halmaza;
- az élek a blokkok végpontjából egy másik blokk belépési pontjára mutatnak;
- egy csomópontból több él léphet ki;
- egy csomópontba több alapblokkból is el lehet jutni.

Fogalmak:

Gráf	$G = (N, E, n_{entry}, n_{exit})$
$succ(n_i)$	azokank $n_k \in N$ csomópontoknak a halmaza melyekre igaz, hogy $(n_i, n_k) \in E$
$pred(n_i)$	azokank $n_k \in N$ csomópontoknak a halmaza melyekre igaz, hogy $(n_k, n_i) \in E$

Optimalizálás fajtái

- lokális – egy blokkon belül;
- globális – egy eljárás vezérlésfolyam-gráfjára;
- interprocedurális – eljárások között.

Lokális optimalizálás

Ismétlődő kifejezések kiszűrése.

```
t6 = 4*i
x = a[t6]
t7 = 4*i
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = 4*j
a[t10] = x
goto L0
```

Lokális optimalizálás

Ismétlődő kifejezések kiszűrése.

```
t6 = 4*i
x = a[t6]
t7 = 4*i
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = 4*j
a[t10] = x
goto L0
```

Lokális optimalizálás

Ismétlődő kifejezések kiszűrése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = 4*j
a[t10] = x
goto L0
```

Lokális optimalizálás

Ismétlődő kifejezések kiszűrése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = 4*j
a[t10] = x
goto L0
```

Lokális optimalizálás

Ismétlődő kifejezések kiszűrése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = 4*j
a[t10] = x
goto L0
```

Lokális optimalizálás

Ismétlődő kifejezések kiszűrése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = t8
a[t10] = x
goto L0
```

Lokális optimalizálás

Másolatok terjedésének a megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = t8
a[t10] = x
goto L0
```

Lokális optimalizálás

Másolatok terjedésének a megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10 = t8
a[t10] = x
goto L0
```

Lokális optimalizálás

Másolatok terjedésének a megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9
t10 = t8
a[t10] = x
goto L0
```

Lokális optimalizálás

Másolatok terjedésének a megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9
t10 = t8
a[t10] = x
goto L0
```

Lokális optimalizálás

Másolatok terjedésének a megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9
t10 = t8
a[t10] = x
goto L0
```

Lokális optimalizálás

Másolatok terjedésének a megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9
t10 = t8
a[t8] = x
goto L0
```

Lokális optimalizálás

Nem használt kifejezés megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9
t10 = t8
a[t8] = x
goto L0
```

Lokális optimalizálás

Nem használt kifejezés megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9
t10 = t8
a[t8] = x
goto L0
```

Lokális optimalizálás

Nem használt kifejezés megszüntetése.

```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9

a[t8] = x
goto L0
```

Lokális optimalizálás

Nem használt kifejezés megszüntetése.

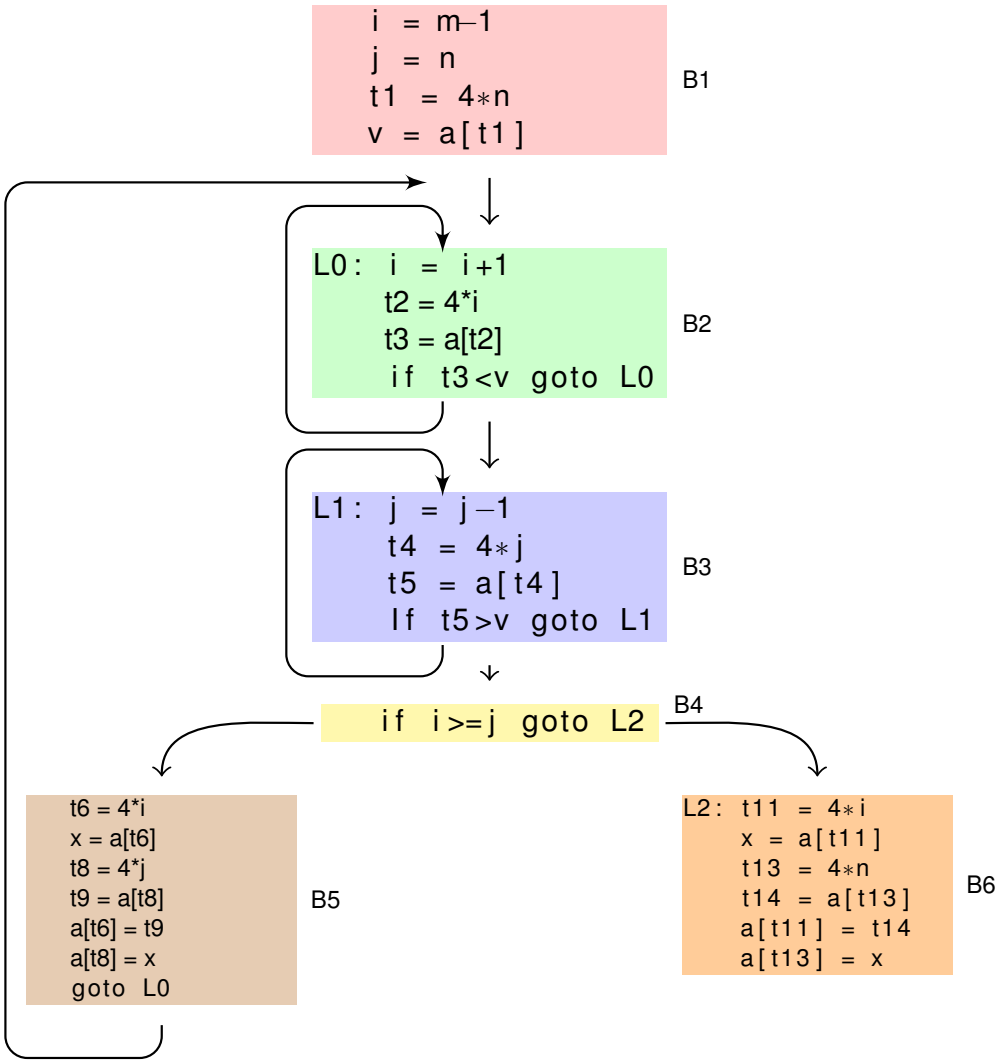
```
t6 = 4*i
x = a[t6]
t7 = t6
t8 = 4*j
t9 = a[t8]
a[t6] = t9

a[t8] = x
goto L0
```

Lokális optimalizálás

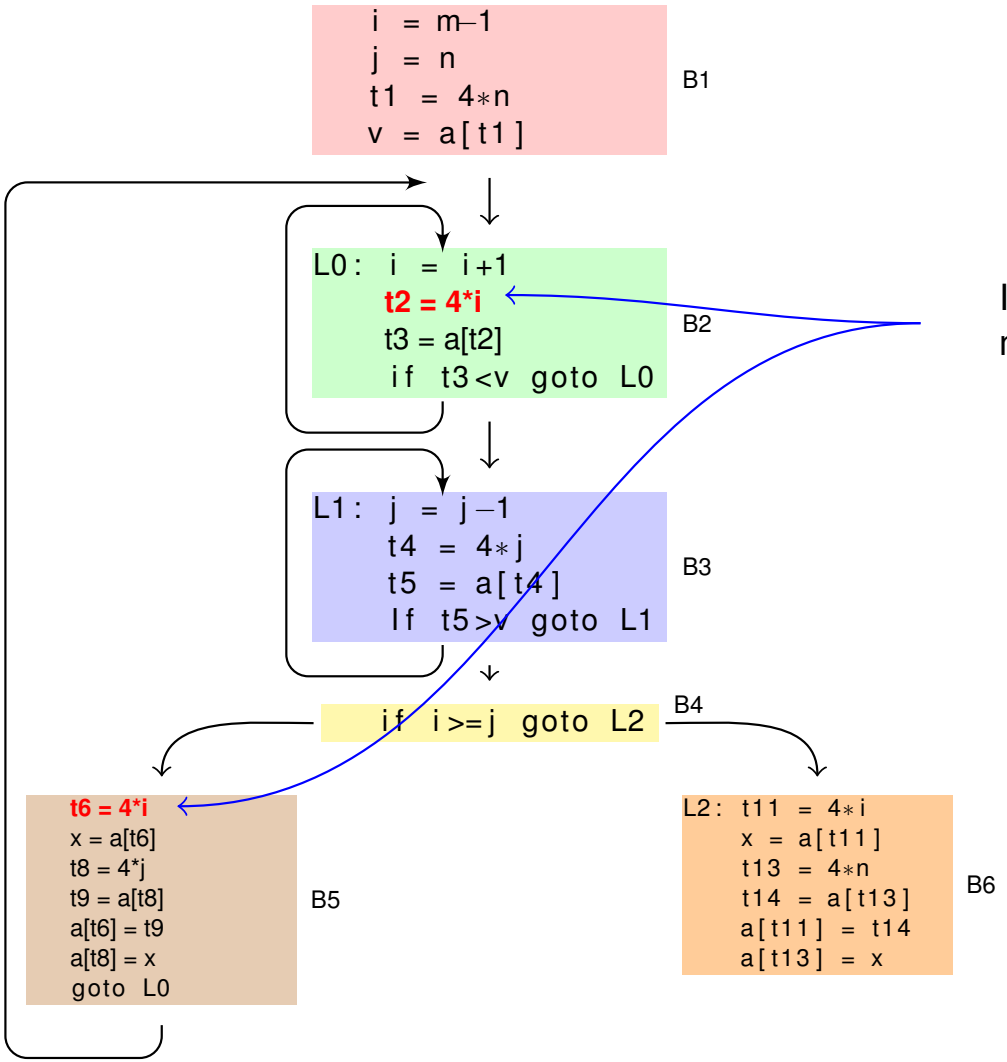
Nem használt kifejezés megszüntetése.

```
t6 = 4*i
x = a[t6]

t8 = 4*j
t9 = a[t8]
a[t6] = t9

a[t8] = x
goto L0
```

Globális optimalizálás

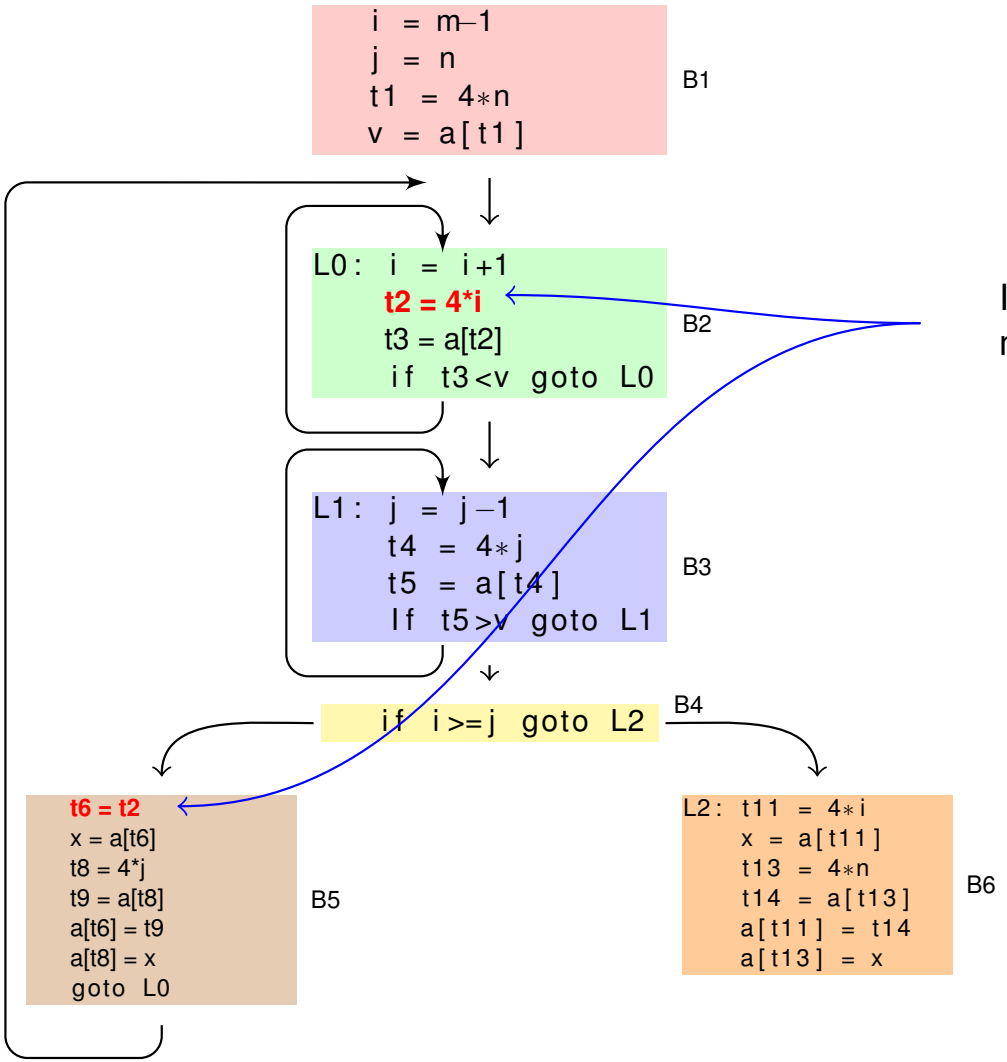


Globális optimalizálás



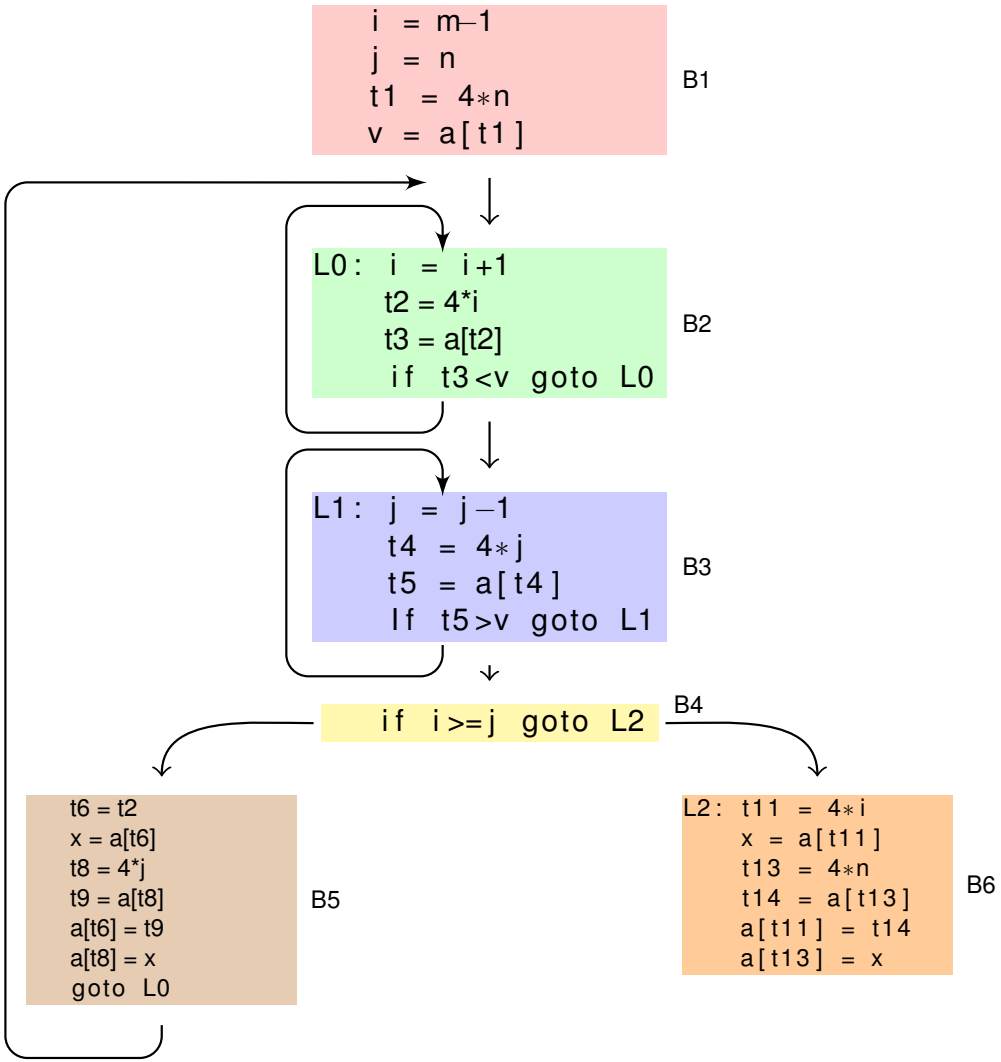
Ismétlődő kifejezések megszüntetése

Globális optimalizálás

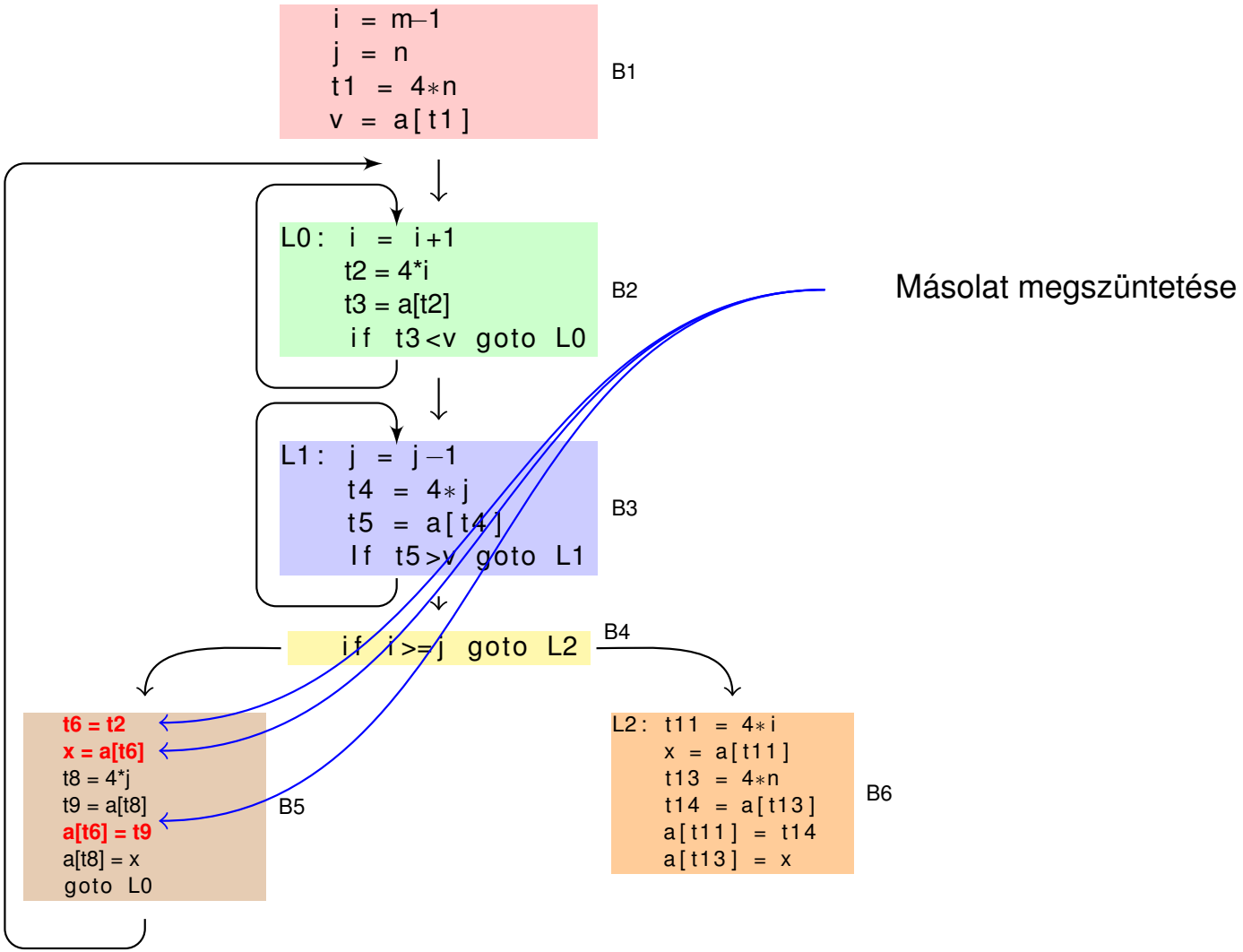


Ismétlődő kifejezések megszüntetése

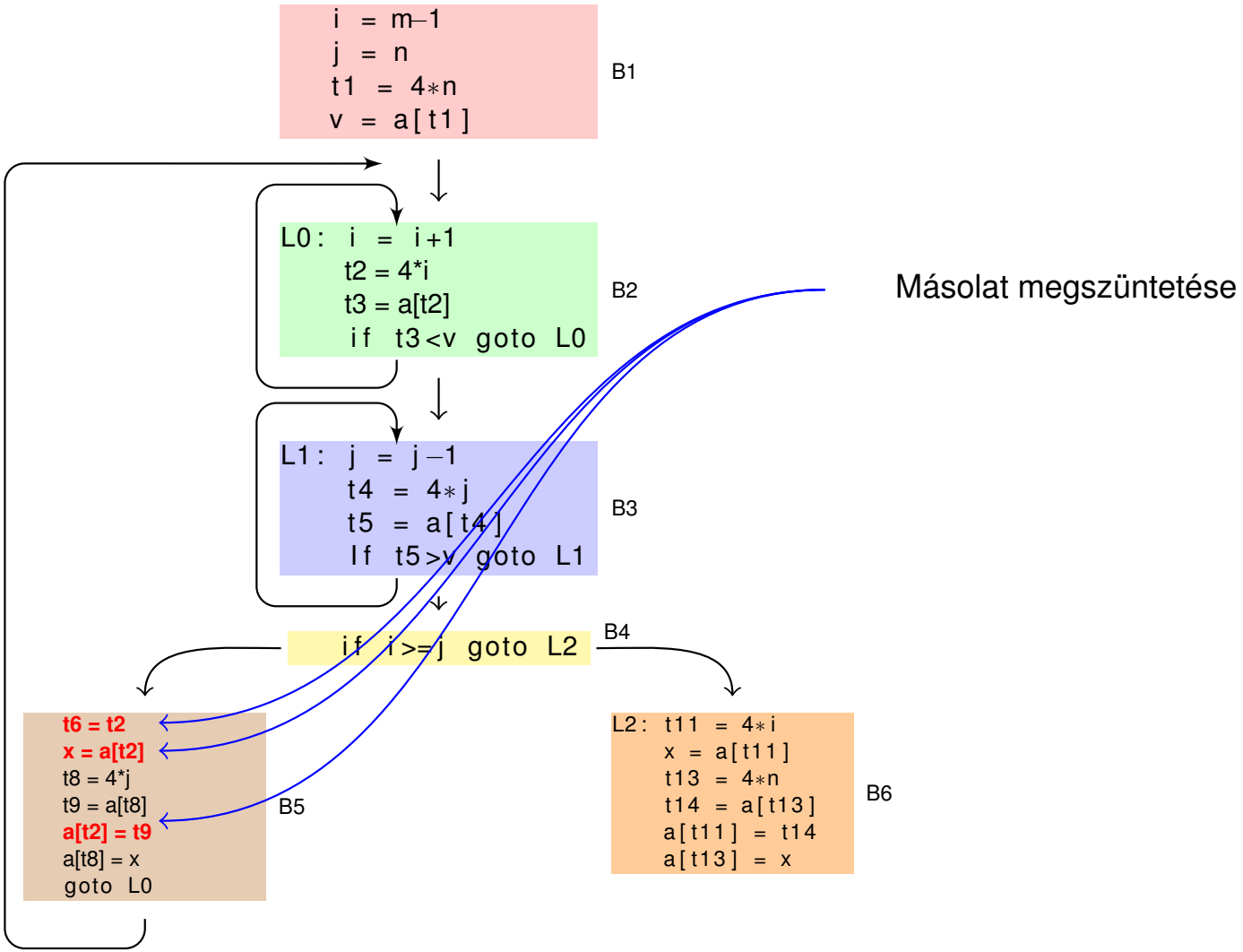
Globális optimalizálás



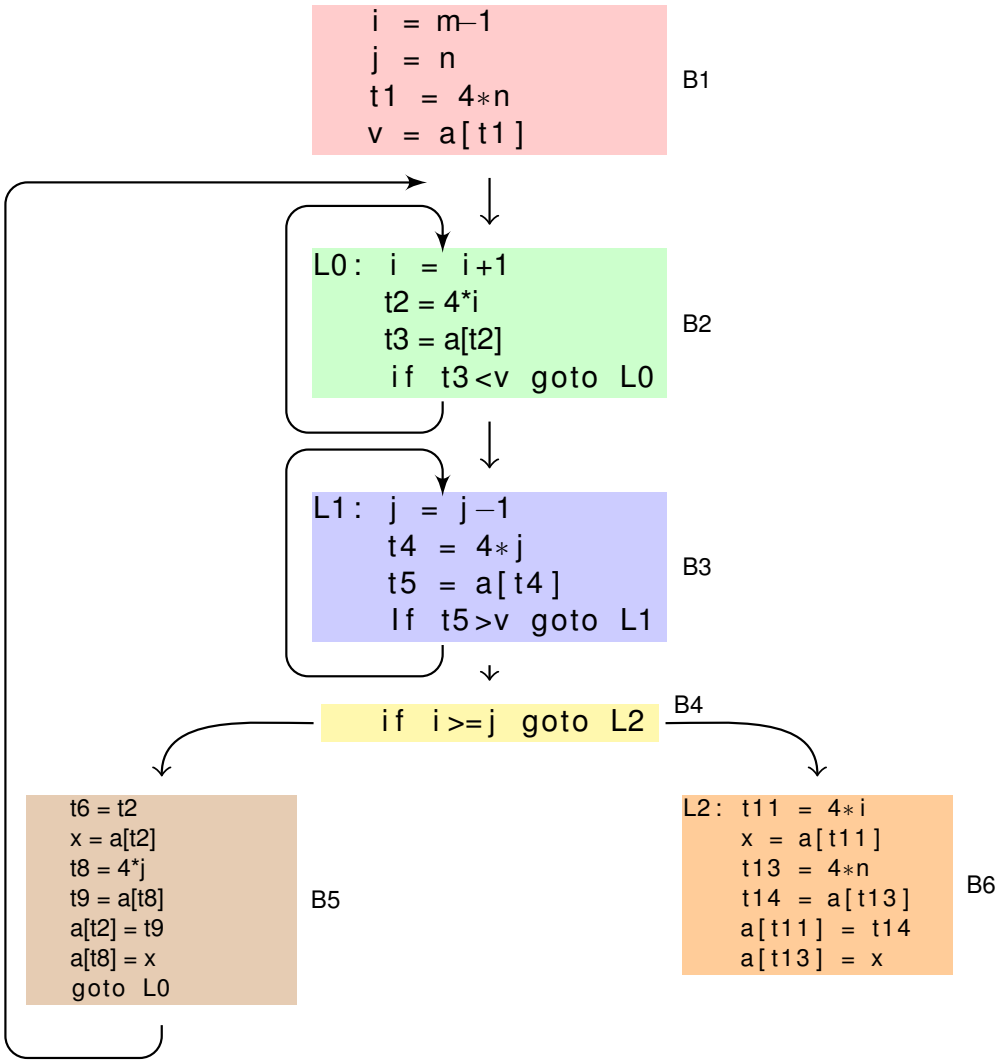
Globális optimalizálás



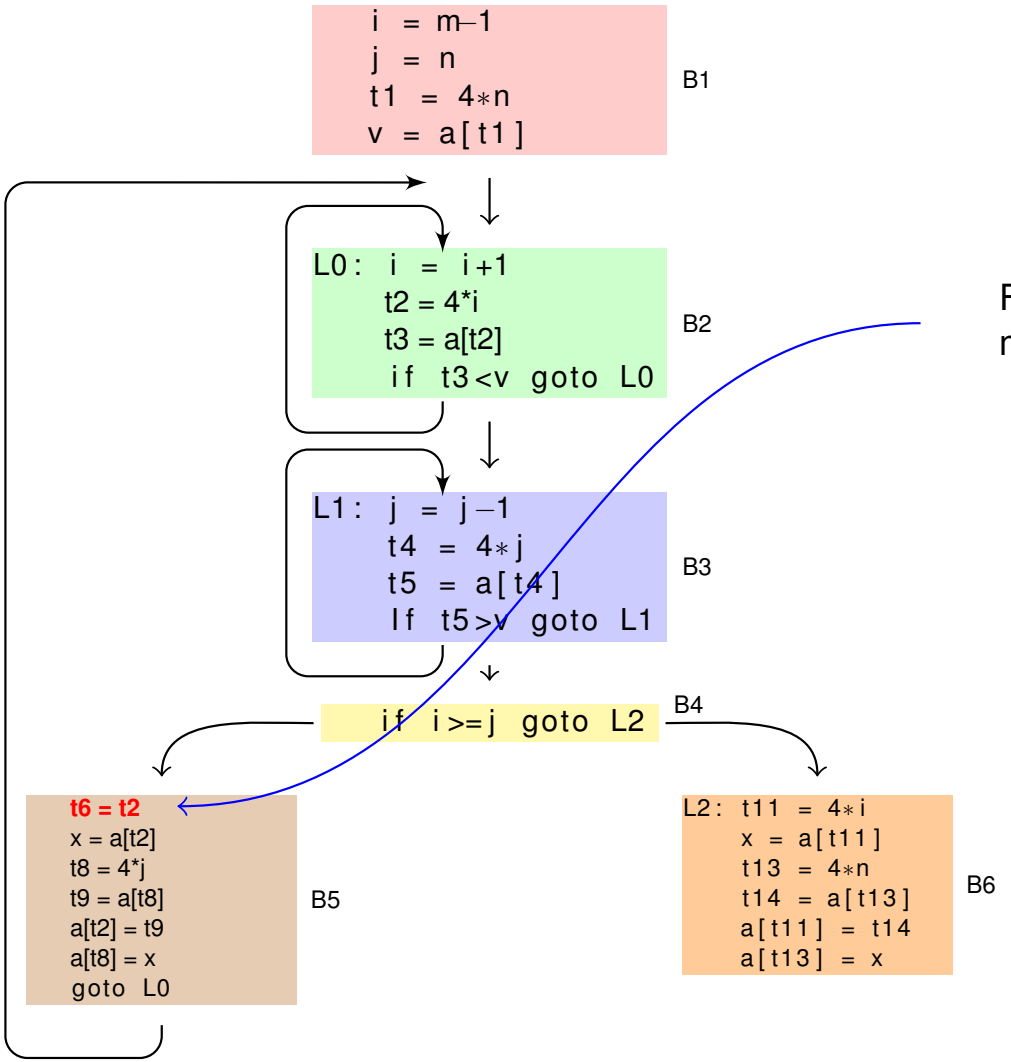
Globális optimalizálás



Globális optimalizálás

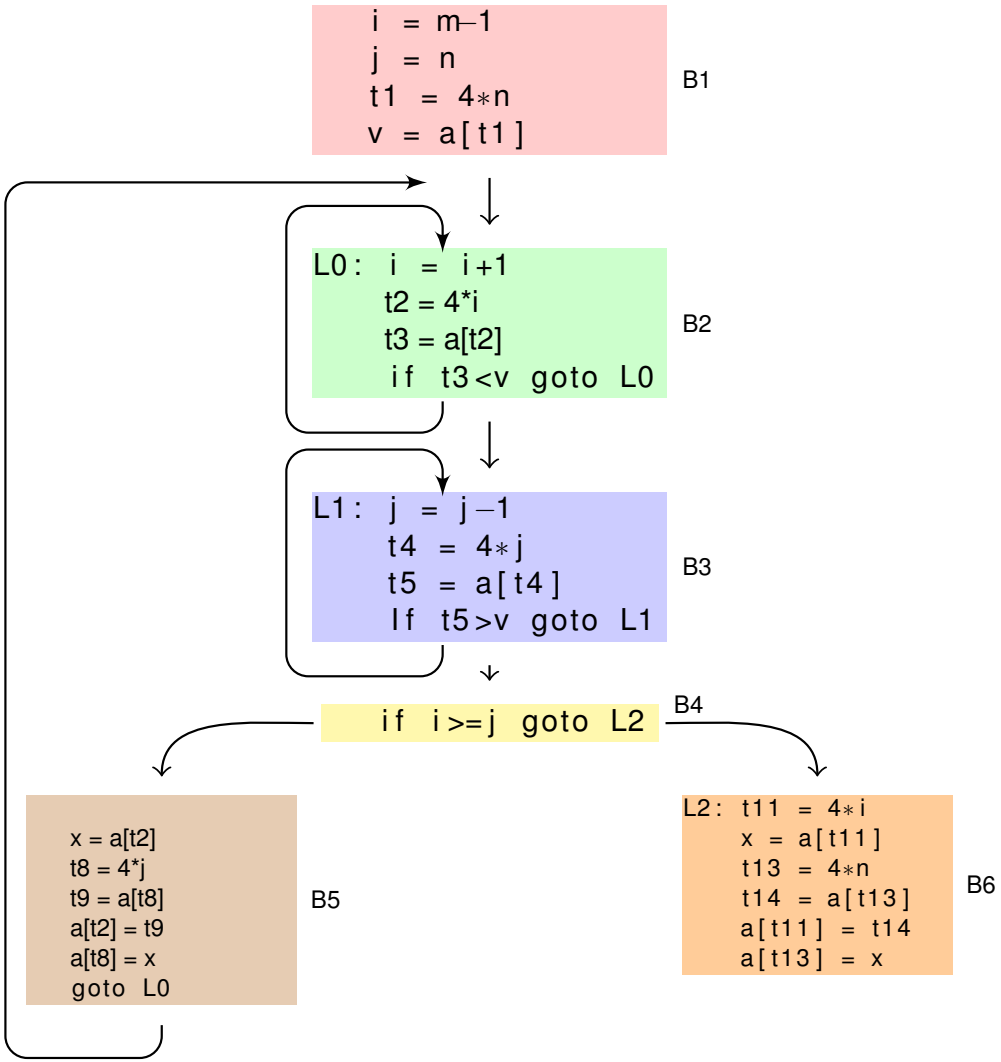


Globális optimalizálás

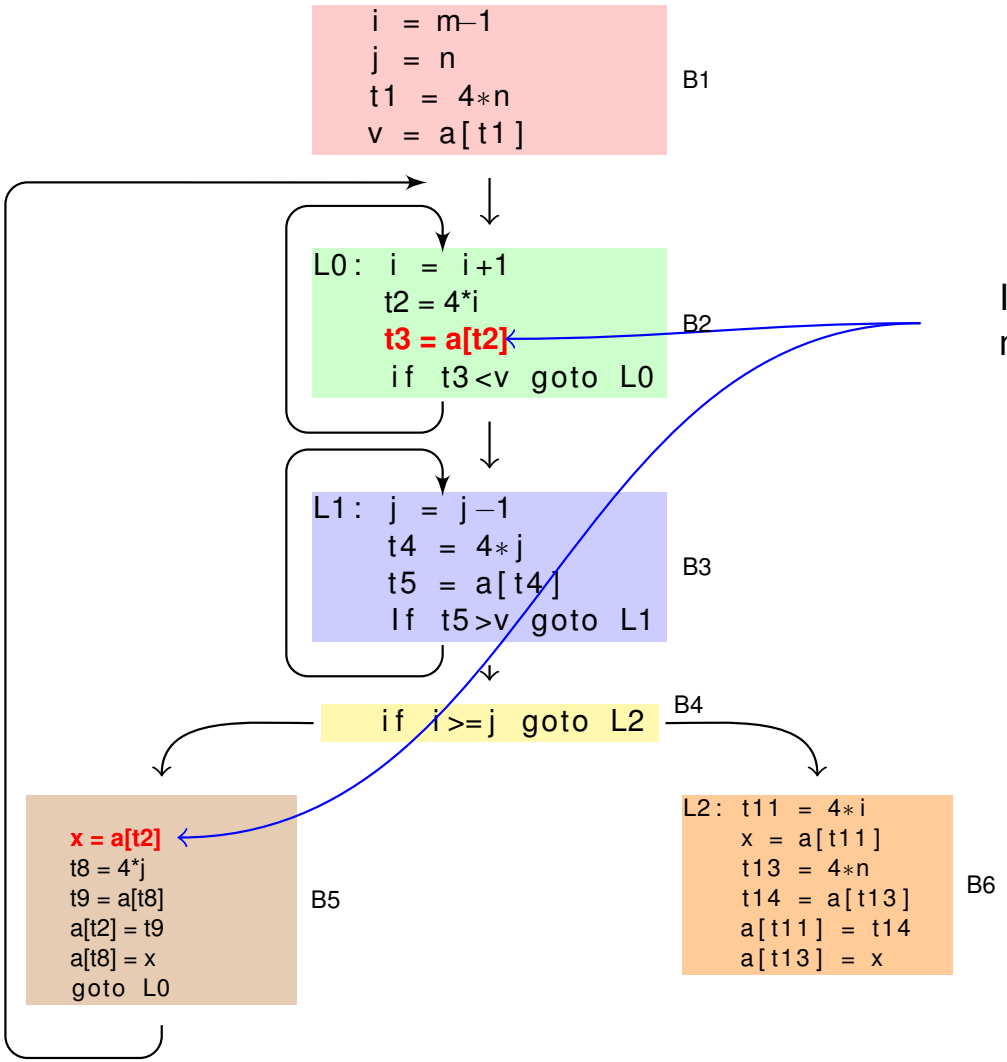


Felesleges kifejezés megszüntetése

Globális optimalizálás

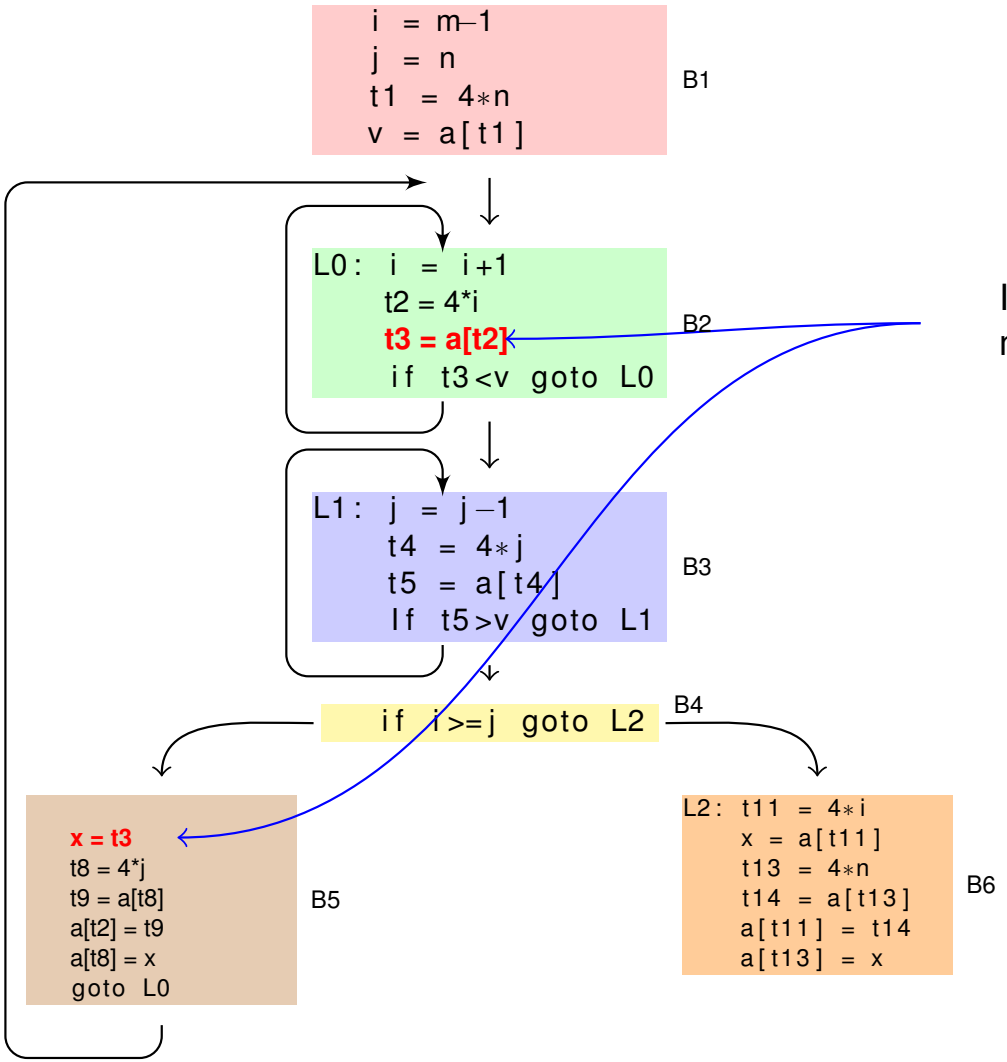


Globális optimalizálás



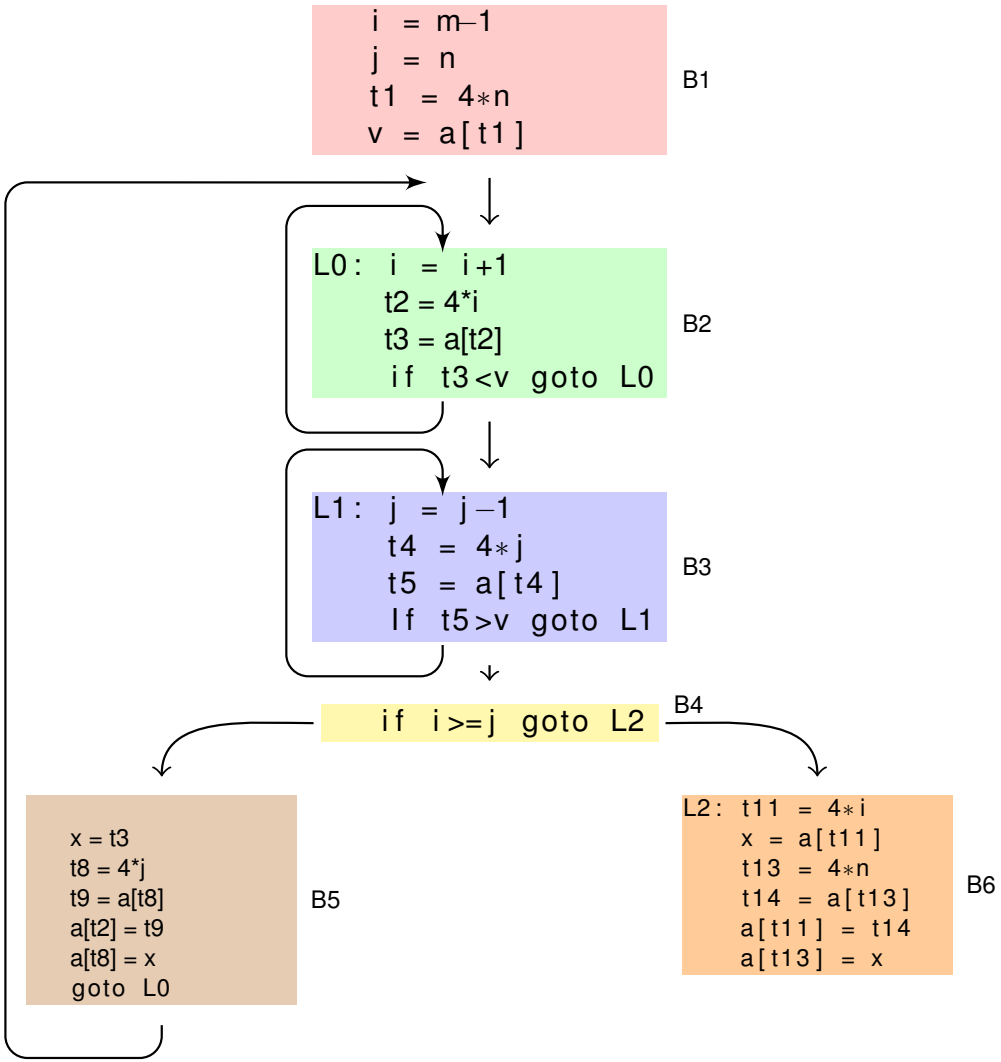
Ismétlődő kifejezések megszüntetése

Globális optimalizálás

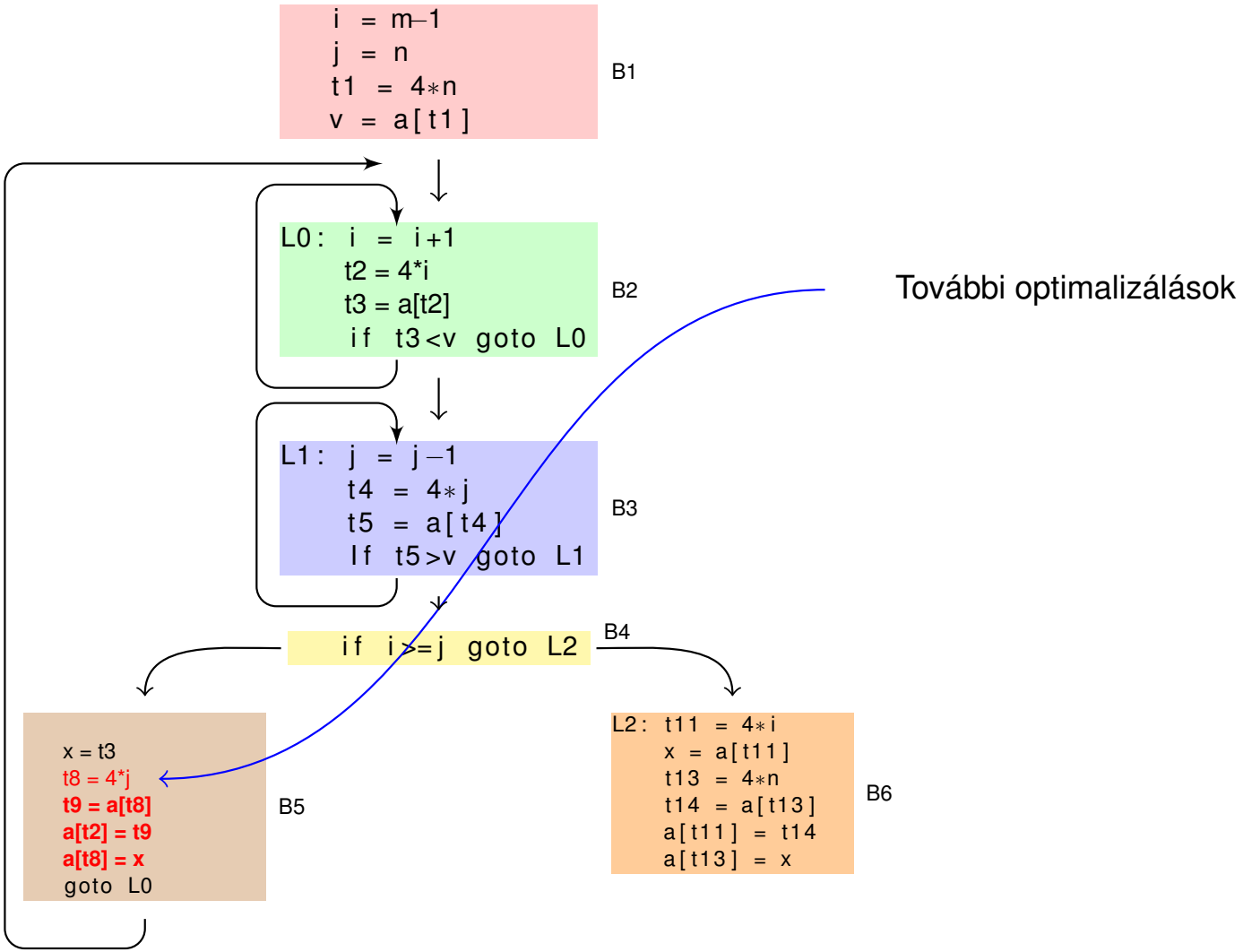


Ismétlődő kifejezések megszüntetése

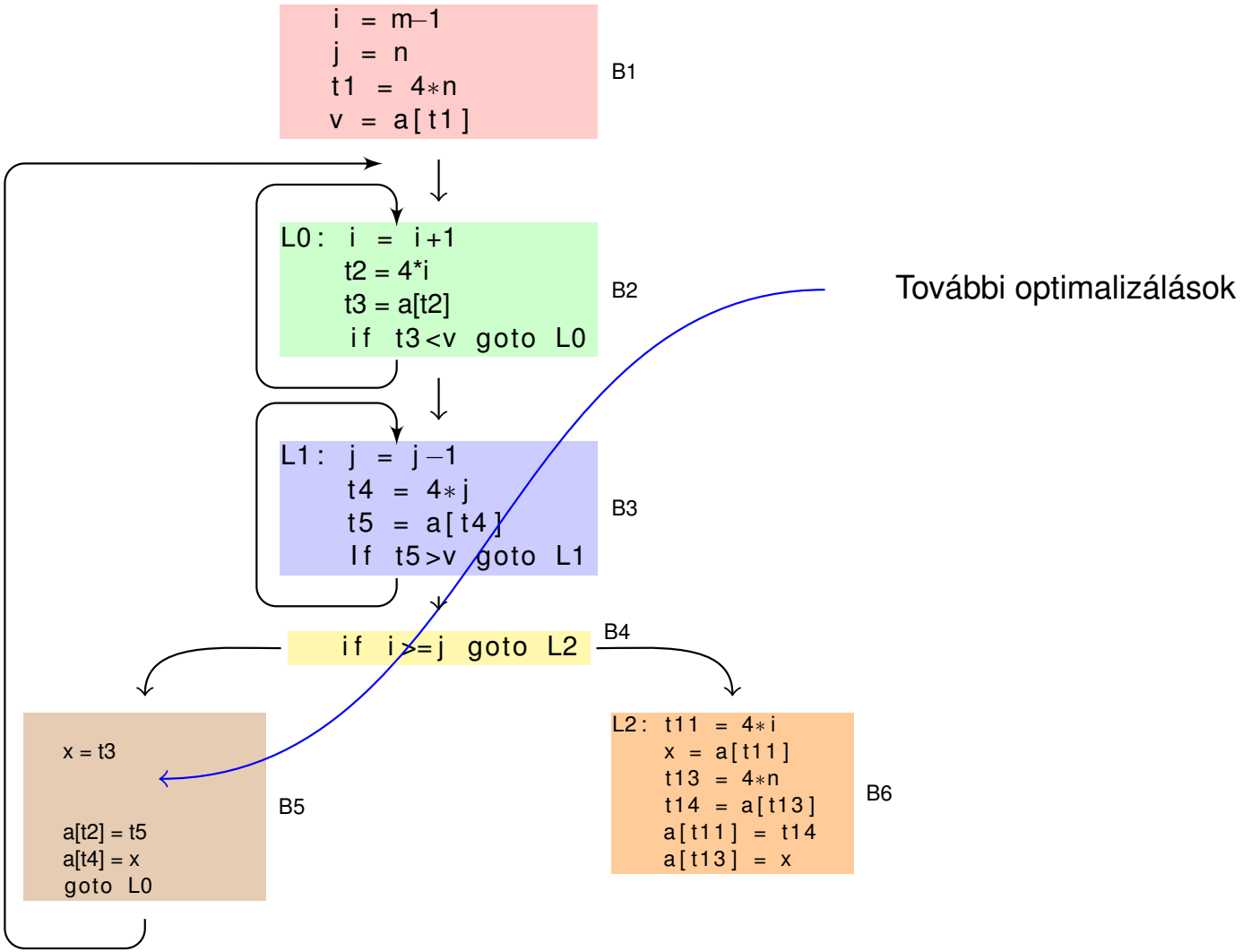
Globális optimalizálás



Globális optimalizálás



Globális optimalizálás



Ismétlődő kifejezések kiszűrése

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = Call Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4 ;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = a + b ;
PushParam t5 ;
PushParam x ;
Call set ;
PopParams 8 ;
```

Ismétlődő kifejezések kiszűrése

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = Call Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4 ;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = a + b ;
PushParam t5 ;
PushParam x ;
Call set ;
PopParams 8 ;
```

Ismétlődő kifejezések kiszűrése

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = Call Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4 ;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = t4 ;
PushParam t5 ;
PushParam x ;
Call set ;
PopParams 8 ;
```

Ismétlődő kifejezések kiszűrése

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = Call Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4 ;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = t4 ;
PushParam t5 ;
PushParam x ;
Call set ;
PopParams 8 ;
```

Ismétlődő kifejezések kiszűrése

- Ha van két értékadás

$$v1 = a\,op\,b$$
$$\ldots$$
$$v2 = a\,op\,b$$

és az a **és** b **operandusok értéke közben nem változik**, akkor a következő transzformációt hajthatjuk végre

$$v1 = a\,op\,b$$
$$\ldots$$
$$v2 = v1$$

- Ezzel elhagyjuk az ismétlődő számításokat.
- lehetőséget teremtünk egy későbbi optimalizáláshoz.

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = t4;
PushParam t5 ;
PushParam x ;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = t4;
PushParam t5 ;
PushParam x ;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = t4;
PushParam t5 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = a + b ;
c = t4 ;
t5 = t4;
PushParam t5 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = t3 + b ;
c = t4 ;
t5 = t4;
PushParam t5 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = t3 + b ;
c = t4 ;
t5 = t4;
PushParam  t5 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = t3 + b ;
c = t4 ;
t5 = t4;
PushParam  t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = t3 ;
t4 = t3 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = c;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Másolat továbbterjedés

- angol szakirodalomban a Copy Propagation kifejezés él;
- ha lehet, akkor máshol is az eredeti példányra szeretnénk hivatkozni;
- Ha van egy értékadás

$$v1 = v2$$

és a továbbiakban **a** $v1$ **és** $v2$ **operandusok értéke nem változik**, akkor a következő transzformációt hajthatjuk végre

$$a = \cdots v1 \cdots$$

lecseréljük:

$$a = \cdots v2 \cdots$$

- Ez a transzformáció később a segítségünkre lesz.

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
x = t1 ;
t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

t3 = 4;
a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

a = 4 ;
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;
```

```
t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

t4 = 4 + b ;
c = t4 ;
t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

t4 = 4 + b ;

t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

t4 = 4 + b ;

t5 = t4;
PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése (Dead Code Elimination)

```
int* x;
int a;
int b;
int c;
x = new int(2);
a = 4;
c = a + b;
set(x,a + b);
```

```
t0 = 4 ;
PushParam t0 ;
t1 = LCall Alloc ;
PopParams 4 ;
t2 = 2 ;
*(t1) = t2 ;

t4 = 4 + b ;

PushParam t4 ;
PushParam t1;
Call set;
PopParams 8 ;
```

Felesleges utasítások törlése

- Értékadás bal oldalán található változó akkor *felesleges*, ha az értékét a későbbiekben nem használjuk fel.
- Ha az értékadás bal oldalán található változó *felesleges*, akkor az egész utasítást el lehet hagyni.

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = a * a;$$
$$d = b + c;$$
$$e = b + b;$$

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = a * a;$$
$$d = b + c;$$
$$e = b + b;$$

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = a * a;$$
$$d = b + c;$$
$$e = b + b;$$

Ismétlődő kifejezések kiszűrése

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + c;$$
$$e = b + b;$$

Ismétlődő kifejezések kiszűrése

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + c;$$
$$e = b + b;$$

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + c;$$
$$e = b + b;$$

Másolat terjedés

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + b;$$
$$e = b + b;$$

Másolat terjedés

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + b;$$
$$e = b + b;$$

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + b;$$
$$e = b + b;$$

Ismétlődő kifejezések kiszűrése

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + b;$$
$$e = d;$$

Ismétlődő kifejezések kiszűrése

Példa lokális optimalizálás végrehajtása

$$b = a * a;$$
$$c = b;$$
$$d = b + b;$$
$$e = d;$$

Iteratív optimalizálási eljárás

A műveleteket addig kell folytatni, amíg a szabályok közül valamelyiket alkalmazni lehet, azaz amíg a fix pontot el nem értük. Mivel a műveletek az eredményt *egy irányba* változtatják, ezért az eljárás mindig leáll.

1. ismétlődő kifejezések kiszűrése;
2. másolat továbbterjedésének a megakadályozása;
3. felesleges utasítások törlése.

Optimalizálás és analízis kapcsolata

Az optimalizálás során szükségünk van a program viselkedésének a megértésére. A **kifejezés elérhetőségének** (available expression) analízise alapján elkerülhető ugyanannak a kifejezésnek a megismételt kiszámítása.

Definició: Elérhető kifejezés

A program adott pontján egy kifejezés akkor érhető el, ha a program kezdetétől az *u* pontig a program minden ágán az adott kifejezés kiszámításra került, és a kiértékelés után a kifejezés egyetlen operandusa sem kapott új értéket.

Példa *elérhető kifejezésre*

Példa:

```
[a:=a+b]1;
[y:=a*b]2;
while [y>a+b]3do
    [a:=a+1]4
    [x:=a+b]5
od
```

A 3-ik címkéjű összehasonlító kifejezésben az `a+b` kifejezés kiértékelése felesleges.

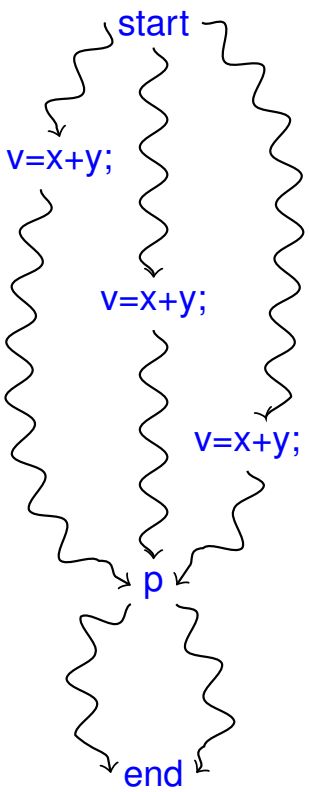
Elérhető kifejezések előállítása

Az *elérhető kifejezés* definíciója nem követeli meg, hogy a kifejezés értékét a program minden ágán ugyanaz a változó hordozza, illetve hogy az értéket hordozó változó később ne kapjon értéket. Temporális változó használatával a kifejezés újabb kiértékelése elkerülhető.

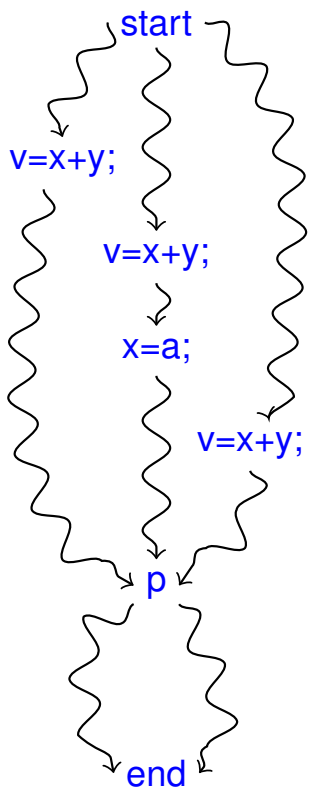
- Az n_{entry} alapblokk elején az *elérhető kifejezés*ek halmaza üres.
- Amikor egy **a = b + c** kifejezéshez érünk:
  - Minden kifejezést, amely az **a**-t tartalmazza, ki kell venni a halmazból.
  - A **b + c** kifejezést be kell tenni az elérhető kifejezések halmazába.
- Mit kell tenni, ha az **a = a + b** kifejezéssel találkozunk?

Elérhető kifejezés előállítása

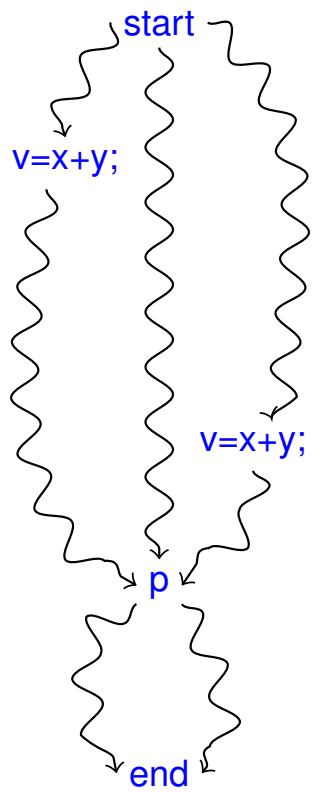
v=x+y a p pontban elérhető



v=x+y a p pontban nem elérhető



v=x+y a p pontban nem elérhető



Elérhető kifejezés előállítása

```
a = b + 2;

c = b / 10;

d = a + b;

e = a + b;

d = b << 2

b = a + b;
```

Elérhető kifejezés előállítása

{}

a = b + 2;

c = b / 10;

d = a + b;

e = a + b;

d = b « 2

b = a + b;

Elérhető kifejezés előállítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

d = a + b;

e = a + b;

d = b « 2

b = a + b;

Elérhető kifejezés előállítása

$\{\}$

a = b + 2;

$\{ b + 2 \}$

c = b / 10;

$\{ b + 2, b / 10 \}$

d = a + b;

e = a + b;

d = b « 2

b = a + b;

Elérhető kifejezés előállítása

$\{\}$

a = b + 2;

$\{ b + 2 \}$

c = b / 10;

$\{ b + 2, b / 10 \}$

d = a + b;

$\{ b + 2, b / 10, a + b \}$

e = a + b;

d = b « 2

b = a + b;

Elérhető kifejezés előállítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

{ b + 2, b / 10 }

d = a + b;

{ b + 2, b / 10, a + b }

e = a + b;

{ b + 2, b / 10, a + b}

d = b « 2

b = a + b;

Elérhető kifejezés előállítása

$\{\}$

a = b + 2;

$\{ b + 2 \}$

c = b / 10;

$\{ b + 2, b / 10 \}$

d = a + b;

$\{ b + 2, b / 10, a + b \}$

e = a + b;

$\{ b + 2, b / 10, a + b\}$

d = b « 2

$\{ b + 2, b / 10, b « 2, a + b \}$

b = a + b;

Elérhető kifejezés előállítása

$\{\}$

a = b + 2;

$\{ b + 2 \}$

c = b / 10;

$\{ b + 2, b / 10 \}$

d = a + b;

$\{ b + 2, b / 10, a + b \}$

e = a + b;

$\{ b + 2, b / 10, a + b\}$

d = b « 2

$\{ b + 2, b / 10, b \ll 2, a + b \}$

b = a + b;

$\{ a + b\}$

Azonos kifejezések eltávolítása

$\{\}$

a = b + 2;

$\{ b + 2 \}$

c = b / 10;

$\{ b + 2, b / 10 \}$

d = a + b;

$\{ b + 2, b / 10, a + b \}$

e = a + b;

$\{ b + 2, b / 10, a + b \}$

d = b « 2

$\{ b + 2, b / 10, b « 2 , a + b \}$

b = a + b;

$\{ a + b\}$

Azonos kifejezések eltávolítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

{ b + 2, b / 10 }

d = a + b;

{ b + 2, b / 10, a + b }

e = a + b;

{ b + 2, b / 10, a + b }

d = b « 2

{ b + 2, b / 10, b « 2 , a + b }

b = a + b;

{ a + b}

Azonos kifejezések eltávolítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

{ b + 2, b / 10 }

d = a + b;

{ b + 2, b / 10, a + b }

e = d;

{ b + 2, b / 10, a + b }

d = b « 2

{ b + 2, b / 10, b « 2 , a + b }

b = a + b;

{ a + b}

Azonos kifejezések eltávolítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

{ b + 2, b / 10 }

d = a + b;

{ b + 2, b / 10, a + b }

e = d;

{ b + 2, b / 10, a + b }

d = b « 2

{ b + 2, b / 10, b « 2 , a + b }

b = a + b;

{ a + b}

Azonos kifejezések eltávolítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

{ b + 2, b / 10 }

d = a + b;

{ b + 2, b / 10, a + b }

e = d;

{ b + 2, b / 10, a + b }

d = b « 2

{ b + 2, b / 10, b « 2 , a + b }

b = a + b;

{ a + b}

Azonos kifejezések eltávolítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

{ b + 2, b / 10 }

d = a + b;

{ b + 2, b / 10, a + b }

e = d;

{ b + 2, b / 10, a + b }

d = b « 2

{ b + 2, b / 10, b « 2 , a + b }

b = e;

{ a + b}

Azonos kifejezések eltávolítása

{}

a = b + 2;

{ b + 2 }

c = b / 10;

{ b + 2, b / 10 }

d = a + b;

{ b + 2, b / 10, a + b }

e = d;

{ b + 2, b / 10, a + b }

d = b « 2

{ b + 2, b / 10, b « 2 , a + b }

b = e;

{ a + b}

Azonos kifejezések eltávolítása

```
a = b + 2;

c = b / 10;

d = a + b;

e = d;

d = b << 2

b = e;
```

Elérhető kifejezés előállításának a formalizálása

- Jelöljük D-vel a kifejezések halmazát, $e_n \in D$-vel az egyes kifejezéseket, és legyen e_1 az első kifejezés a programban.
- $f_n(x) : x \in D \rightarrow D$ függvény a program n pontjában az elérhető kifejezések halmazához hozzátesz és/vagy elvesz elemeket.
- Gen_n az e_n kifejezést hozzáadja a halmazhoz
- $Kill_n$ kiveszi azokat a kifejezést az x halmazból, melyeknek van olyan operandusa, amely értéke megváltozik az e_n kifejezést tartalmazó utasításban.
- $In_n \subseteq D$ az $f_n(x)$ lehetséges bemenő értéke
- $Out_n = f(In_n)$ ahol $f_n(x) = Gen_n \cup (x - Kill_n)$
-

$$In_n \begin{cases} s_1 \text{esetén} \emptyset, \text{különben az összes kifejezés} & \text{inicializálásnál} \\ \cap_{p \in pred(n)} Out_p, & \text{egyébként} \end{cases}$$

Elérhető kifejezés előállításának a formalizálása

A D halmazt reprezentálhatjuk logikai értékek vektorával jelöljük L-lel.

bit pozició	1	2	3	4	5	6
kifejezés	a=b*10	c=b/2	d=a+b	e=c+b	d=b+d	f=e+b

Például: $\{b * 10, a + b, e + b\} = \langle 101001 \rangle$
Ekkor $f_n(x) = Gen_n \vee (x \wedge \sim Kill_n)$

$$f_3 = f_{d=a+b} = \begin{cases} Gen_3, & \langle 001000 \rangle \\ Kill_3, & \langle 000010 \rangle \end{cases}$$

$$\begin{aligned} f_3(\langle 110010 \rangle) &= \langle 001000 \rangle \vee (\langle 110010 \rangle \wedge \sim \langle 000010 \rangle) \\ &= \langle 001000 \rangle \vee (\langle 110010 \rangle \wedge \langle 111101 \rangle) \\ &= \langle 001000 \rangle \vee \langle 110000 \rangle \\ &= \langle 111000 \rangle \end{aligned}$$

Alapblokk $f_B(x)$ függvényének a kiszámítása

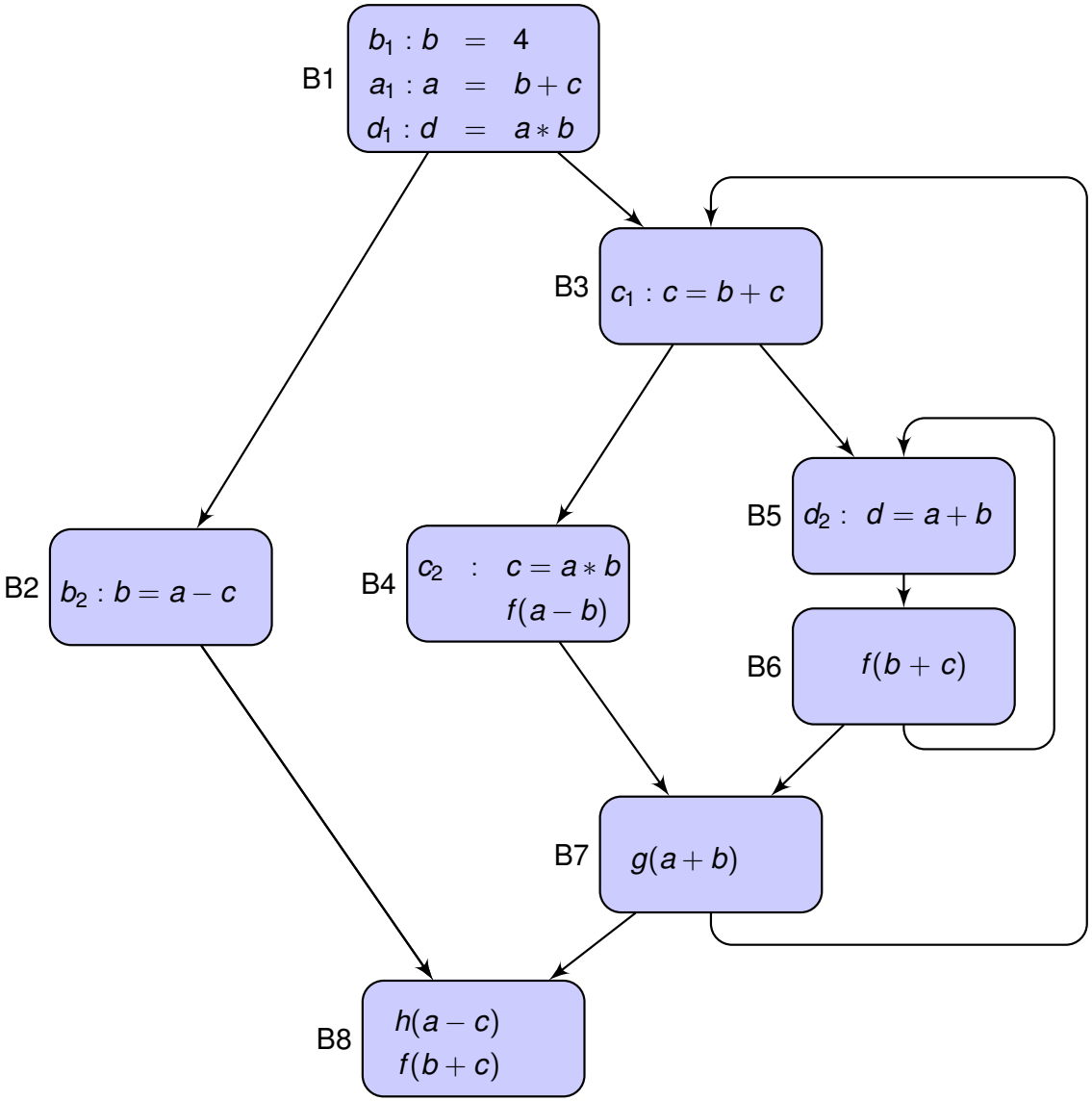
Minden alapblokk $s_1 \ldots s_n$ utasítások szekvenciális sorozatát tartalmazza, ezért az f_B függvény az f_i függvényekből egyszerűen előállítható.

$$\begin{aligned} f_B(x) &= f_n \circ f_{n-1} \ldots f_2 \circ f_1(x) \\ &= f_n(f_{n-1}(\ldots f_2(f_1(x)) \ldots)); \end{aligned}$$

Két utasításra:

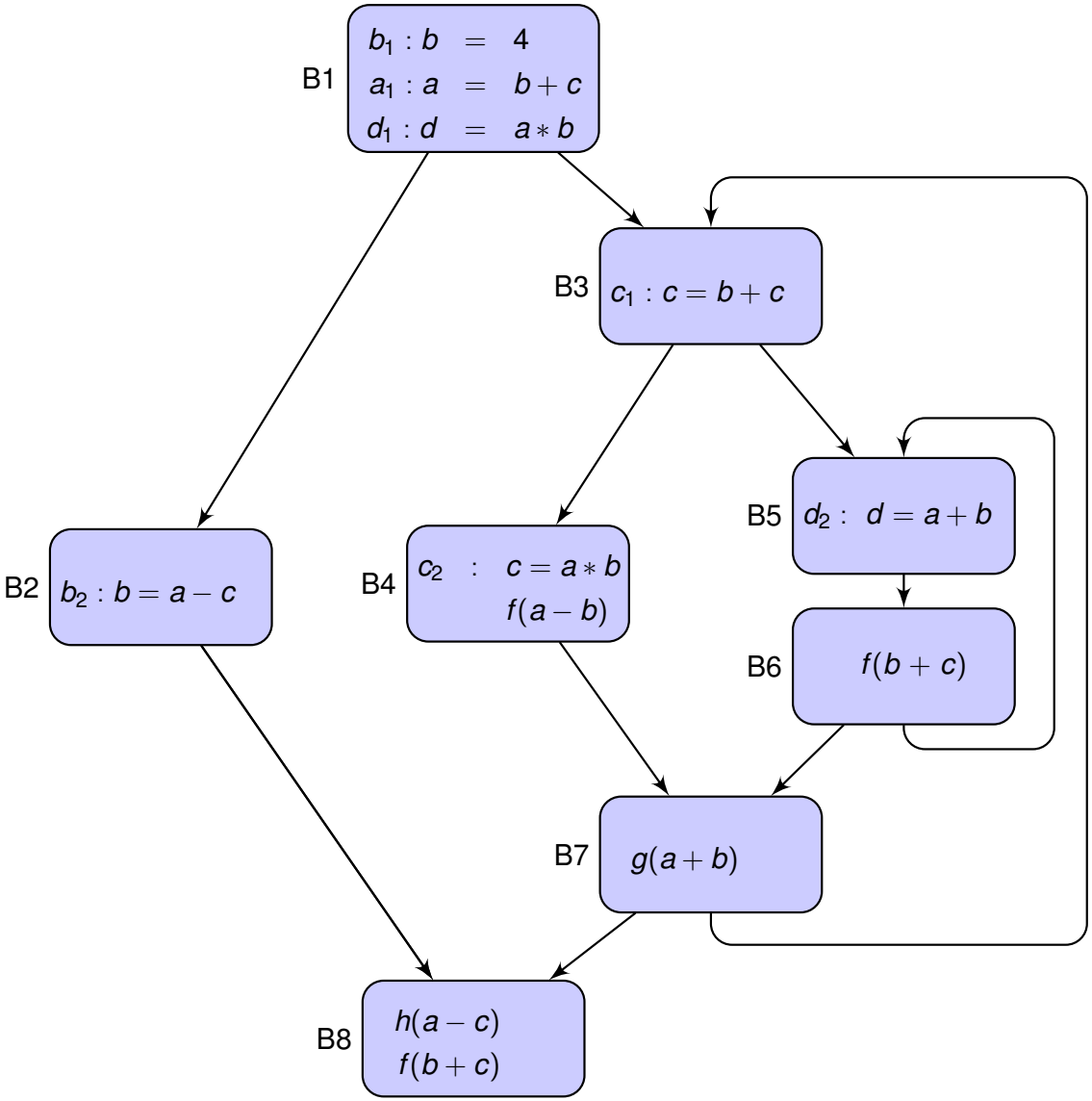
$$\begin{aligned} f_B(x) &= f_2 \circ f_1(x) \\ &= Gen_2 \vee (Gen_1 \vee (x \wedge \sim Kill_1) \wedge \sim Kill_2) \\ &= Gen_2 \vee ((Gen_1 \wedge \sim Kill_2) \vee (x \wedge \sim (kill_1 \vee kill_2)) \end{aligned}$$

Elérhető kifejezések analízise



$$\begin{aligned} \mathbb{V}\text{ar} &= \{a, b, c, d\} \\ \mathbb{D}\text{efs} &= a_1, b_1, b_2, c_1, c_2, d_1, d_2 \\ \mathbb{E}\text{xpr} &= \{a * b, a + b, a - b, a - c, b + c\} \end{aligned}$$

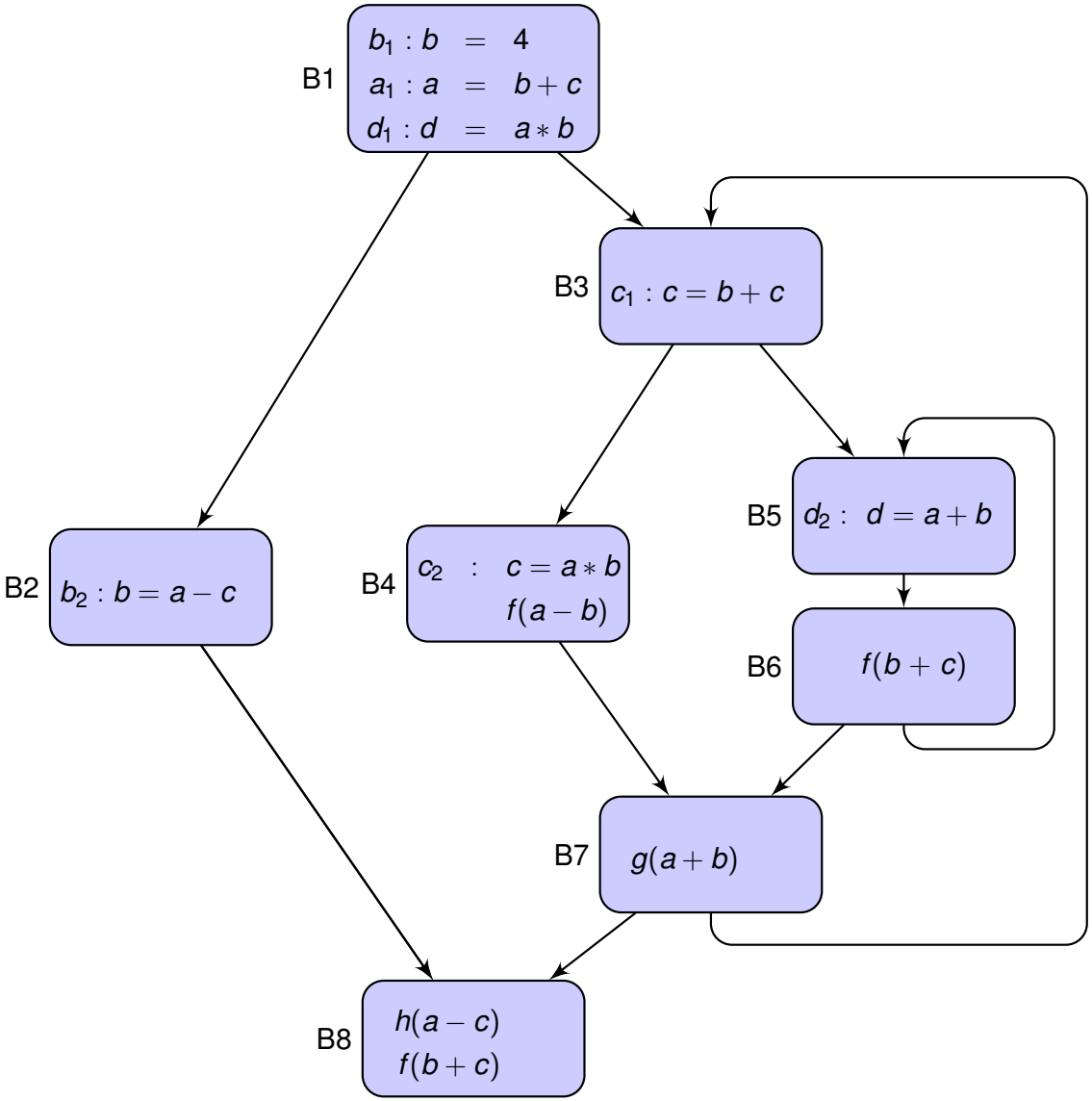
Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

bit pozició:	1	2	3	4	5
kifejezés:	a*b	a+b	a-b	a-c	b+c

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

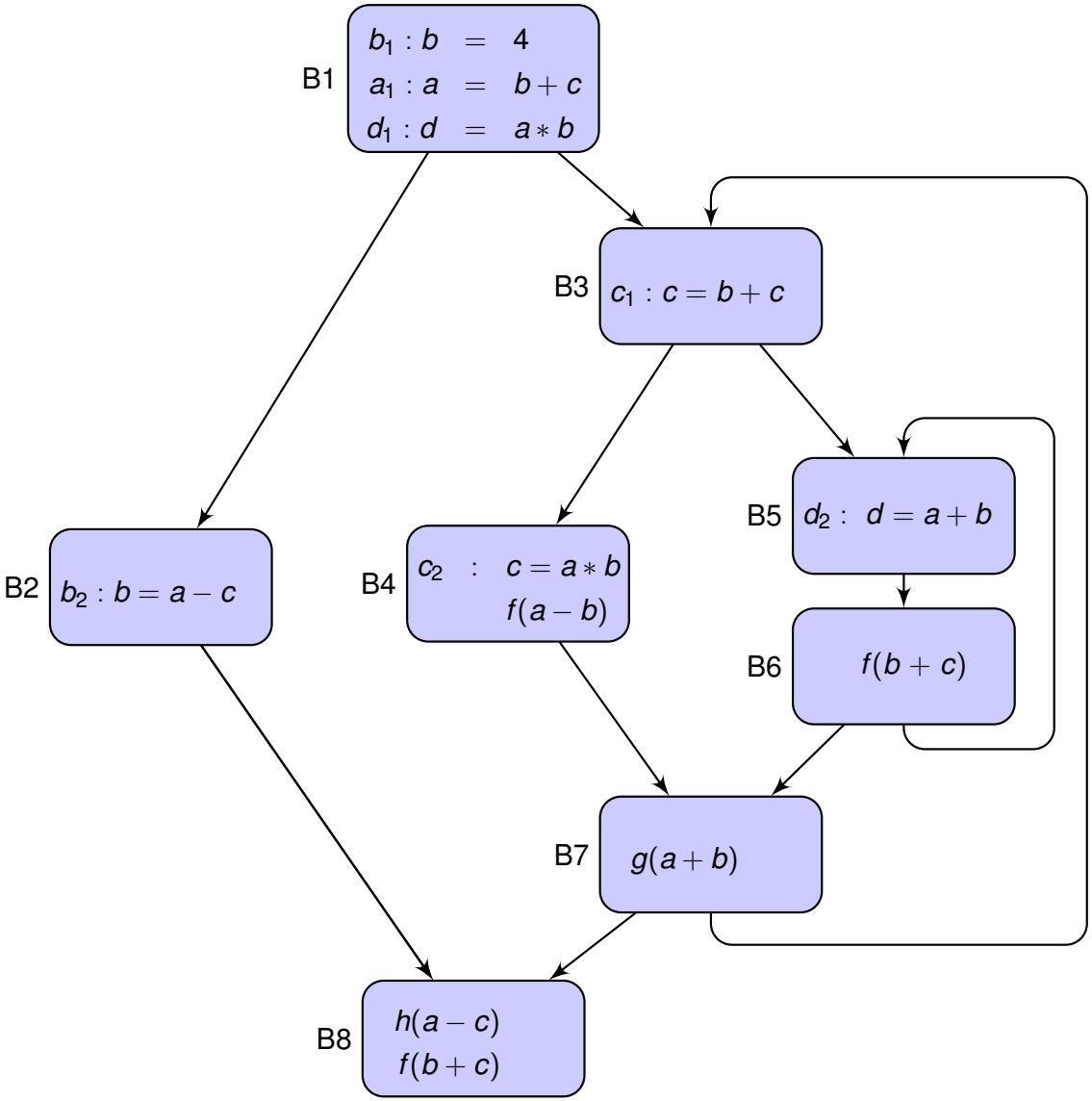
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	11111
B2	00010	11101	00010		11111
B3	00001	00011	00001		11111
B4	10100	00011	10100		11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO:

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

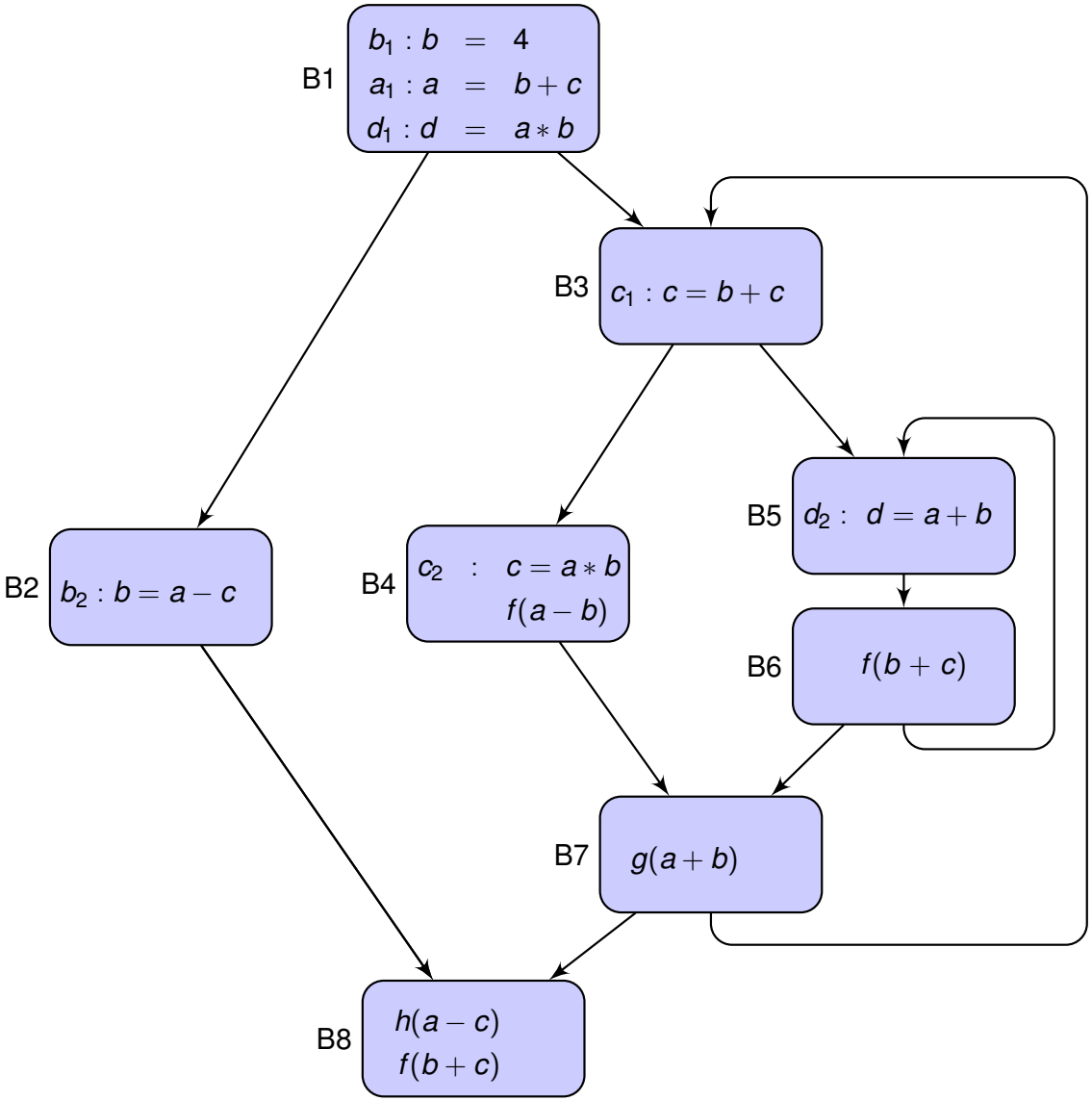
bit pozíció:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	11111
B2	00010	11101	00010		11111
B3	00001	00011	00001		11111
B4	10100	00011	10100		11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B1\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

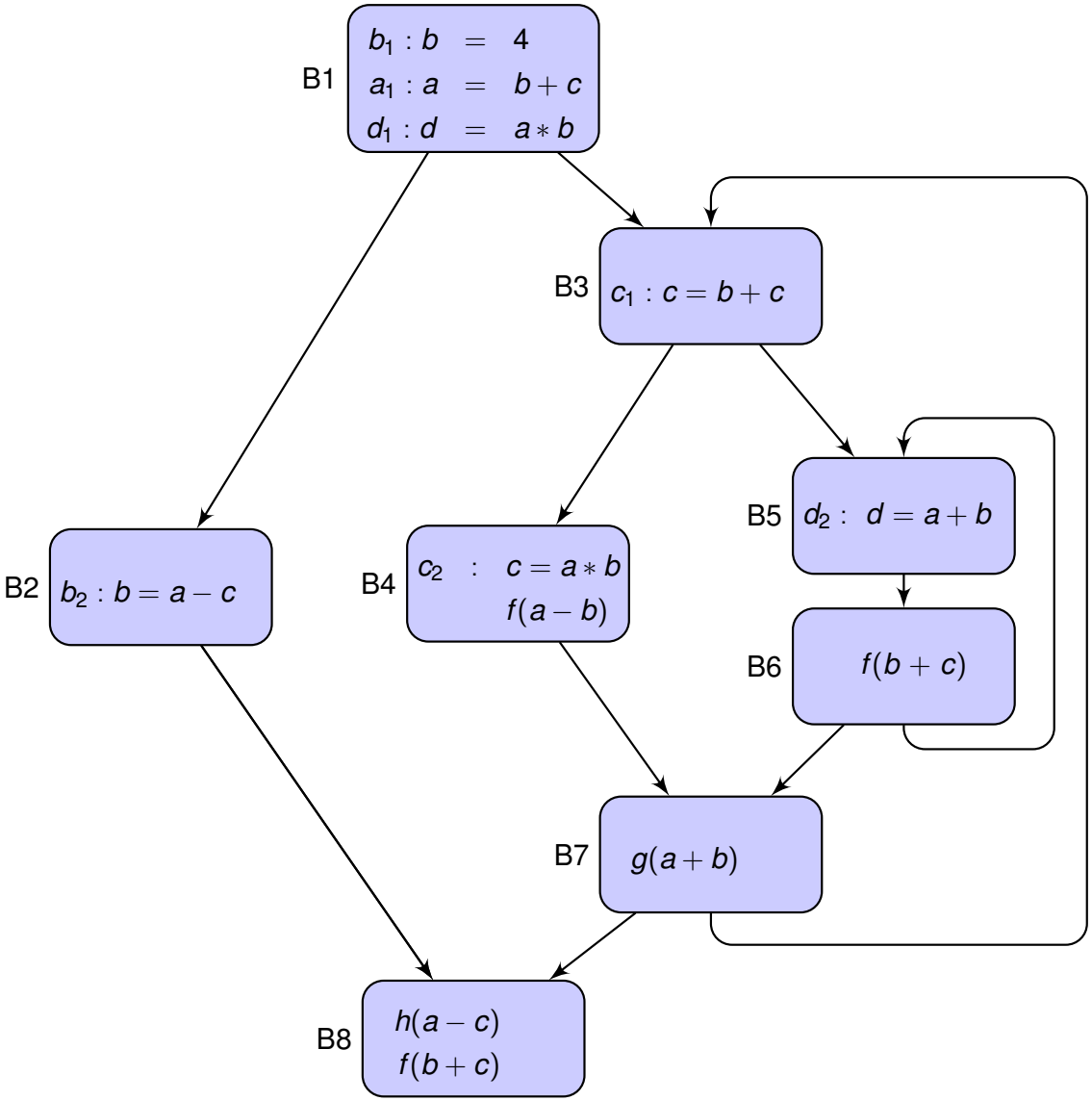
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010		11111
B3	00001	00011	00001		11111
B4	10100	00011	10100		11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B2, B3\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

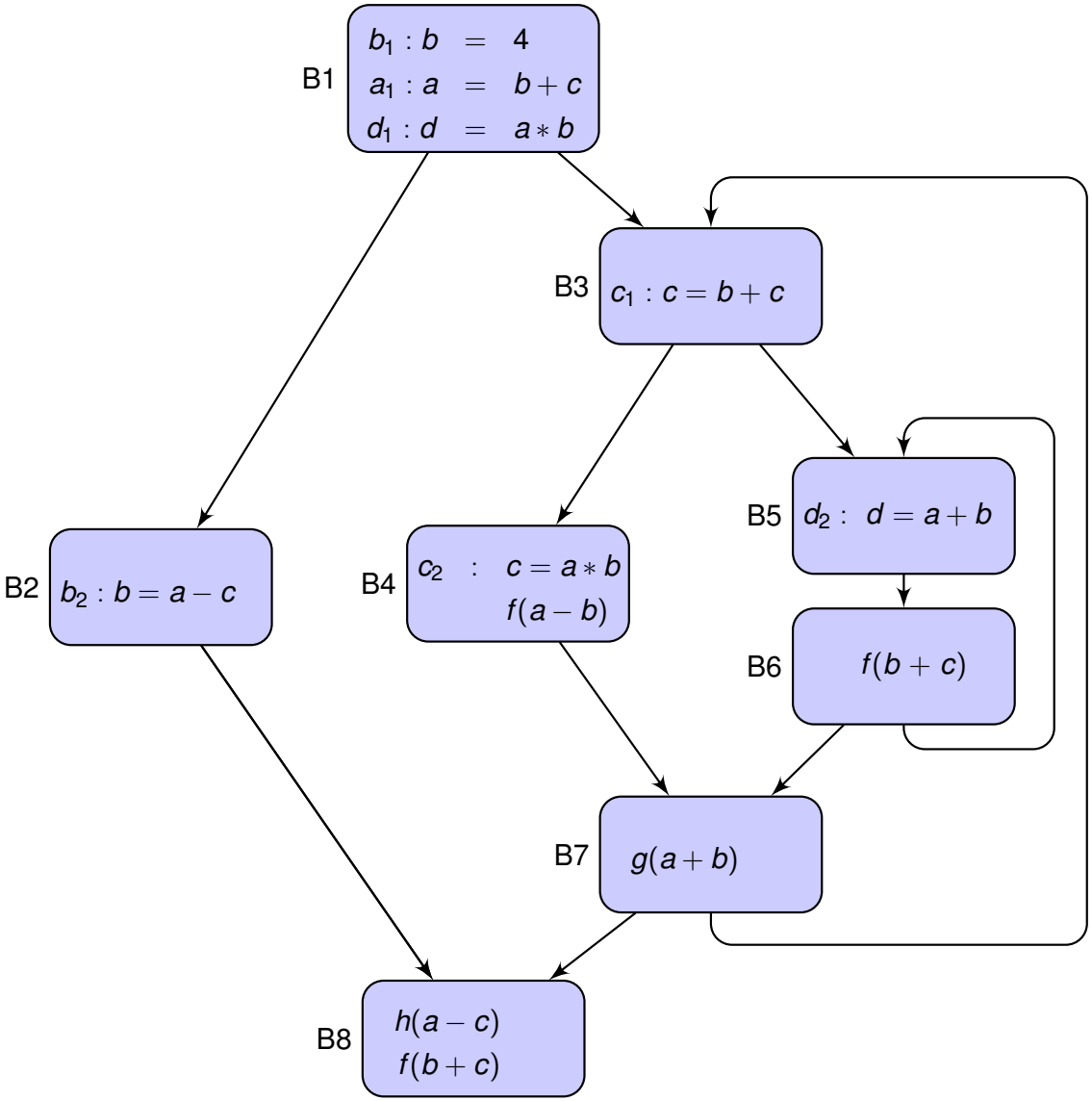
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	11111
B3	00001	00011	00001		11111
B4	10100	00011	10100		11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B2, B3\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

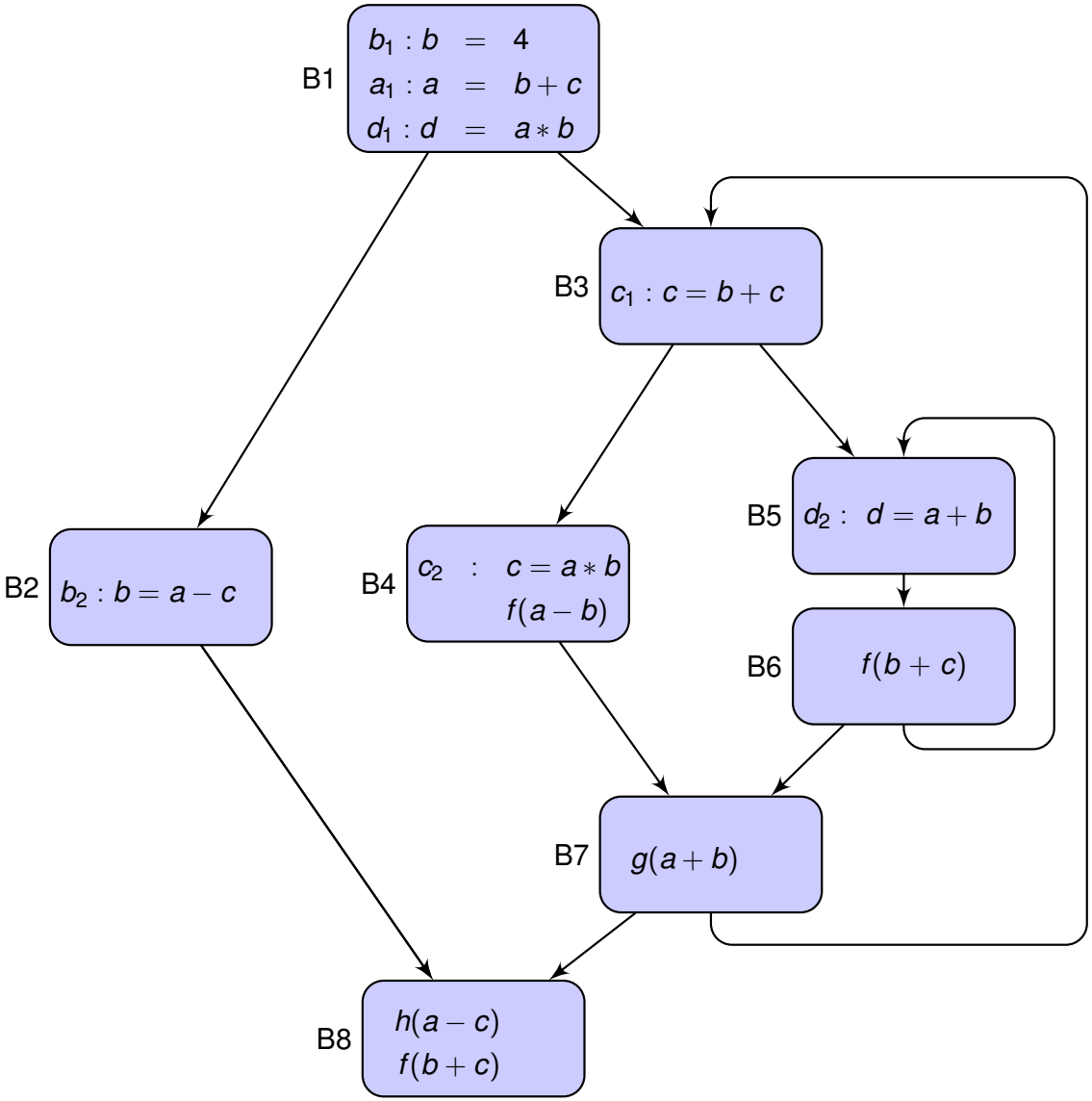
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001		11111
B4	10100	00011	10100		11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B3, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

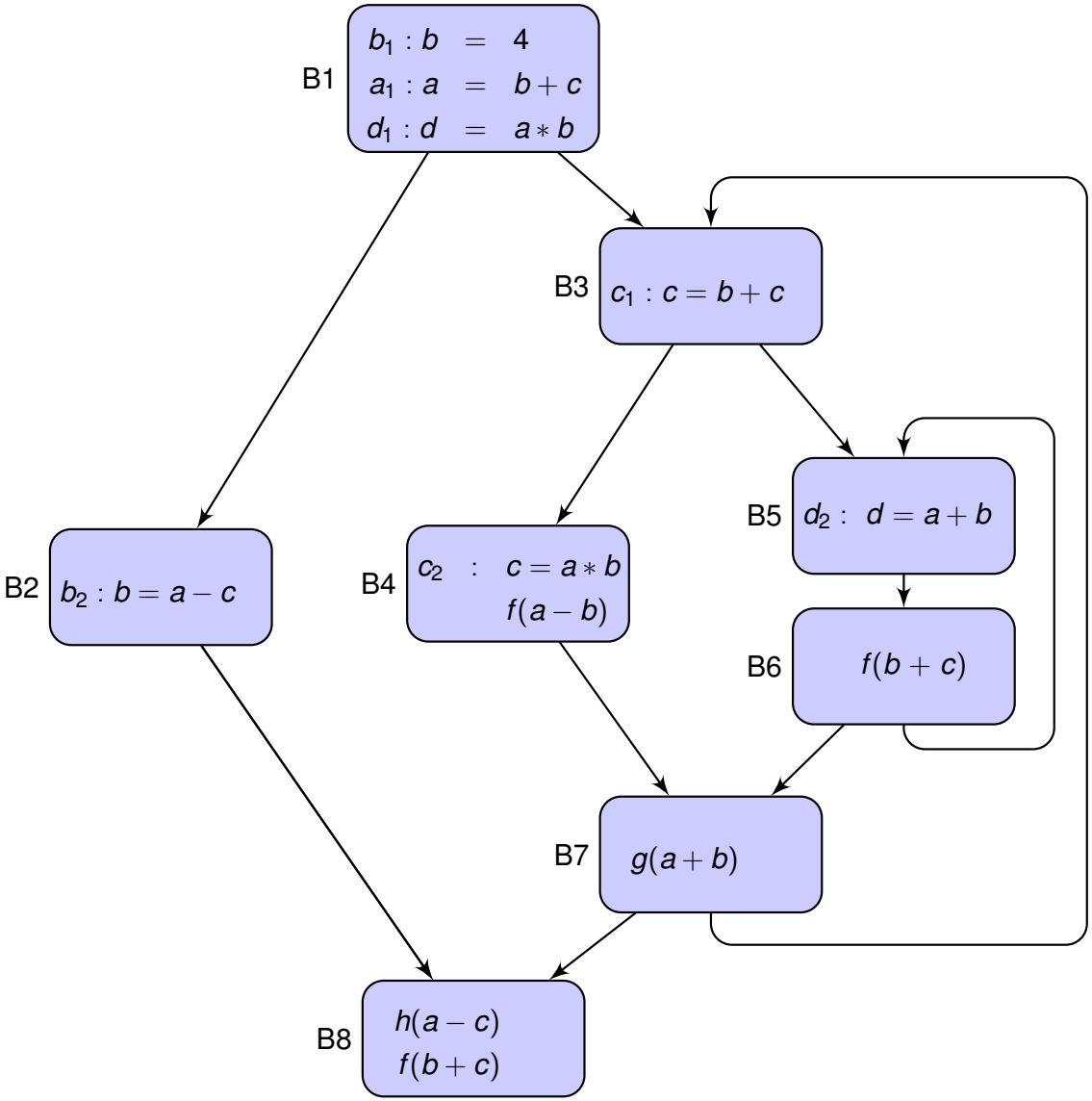
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	11111
B4	10100	00011	10100		11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B3, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

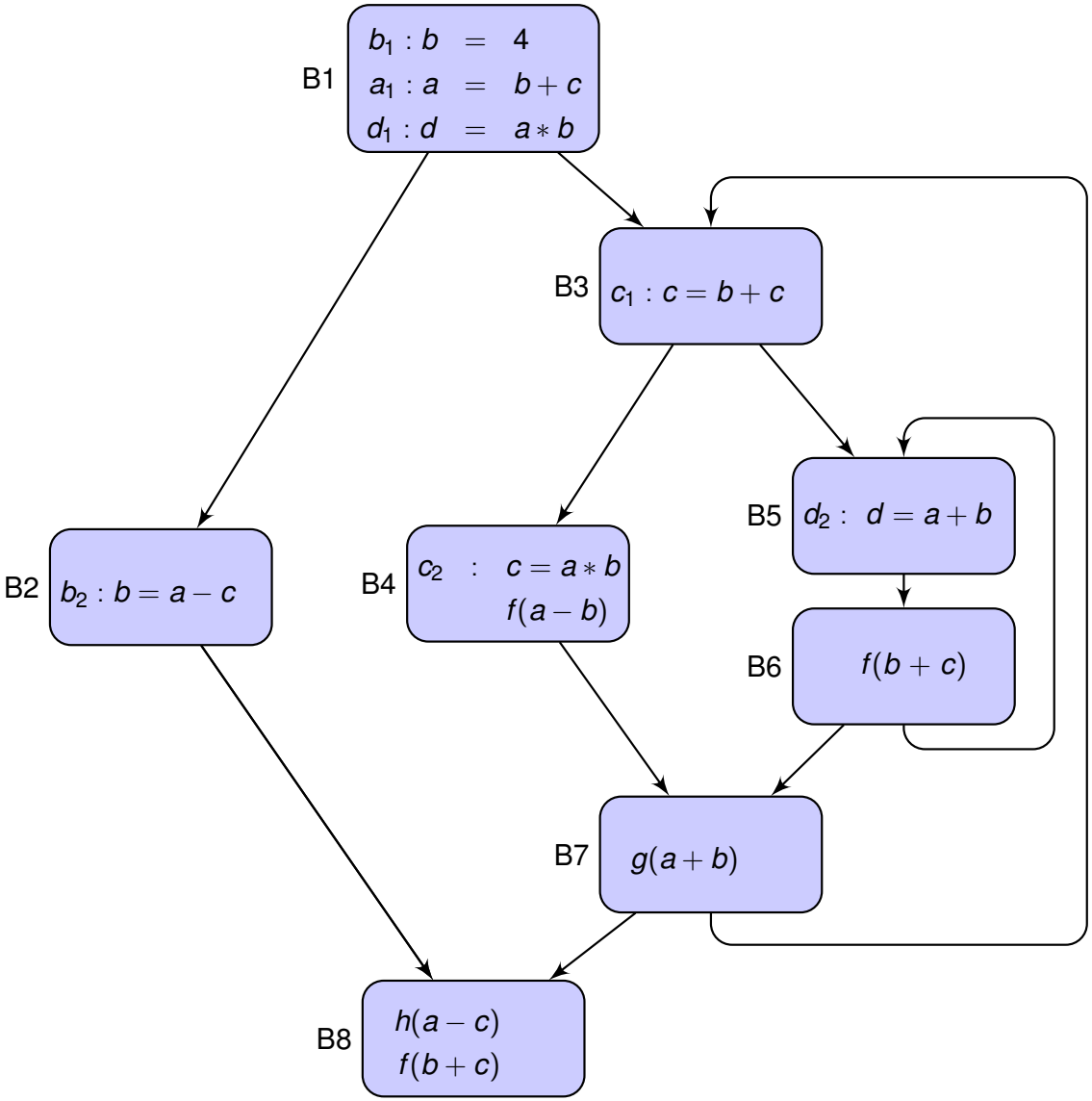
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100		11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B4, B5, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

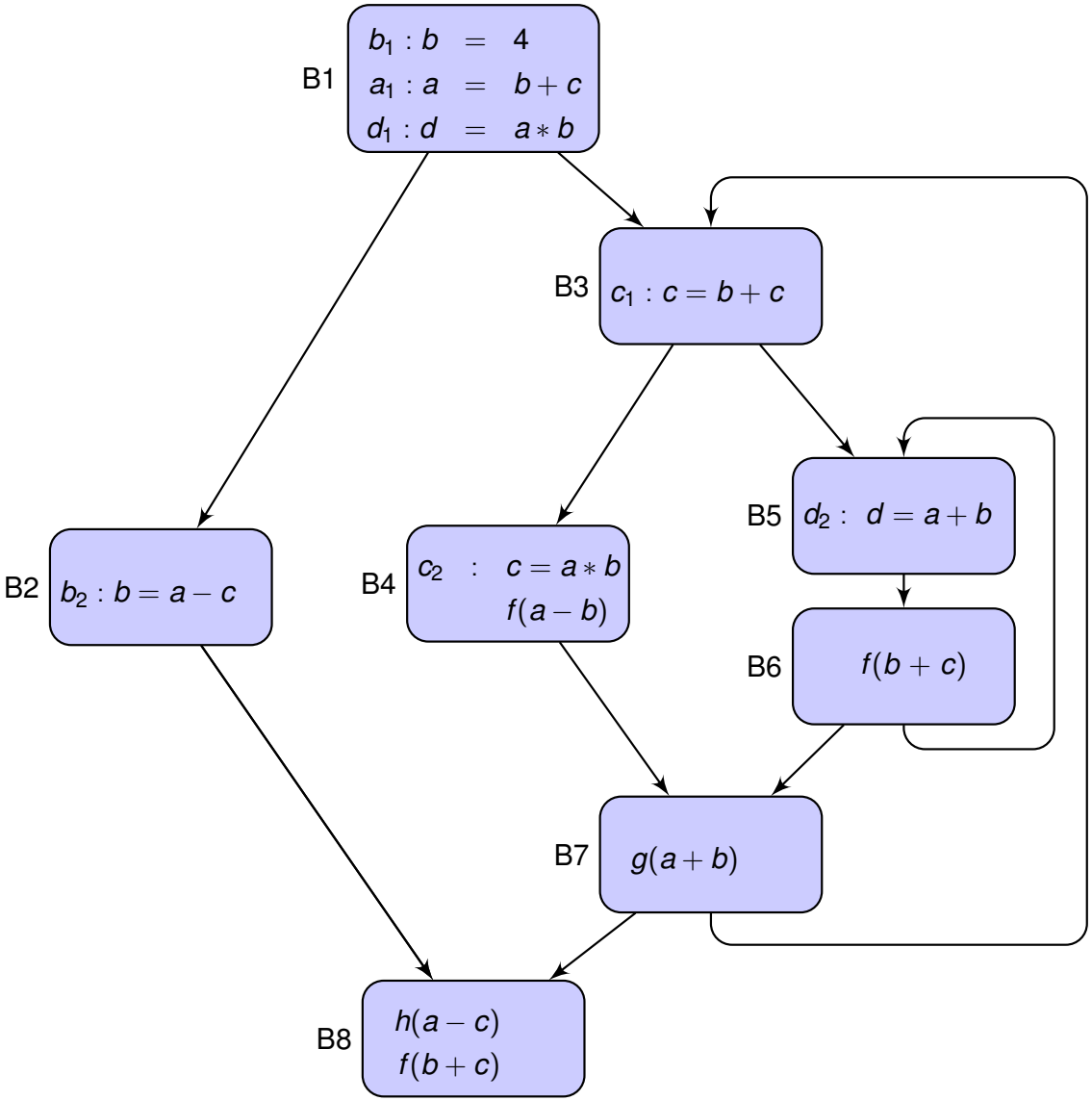
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	11111
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B4, B5, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

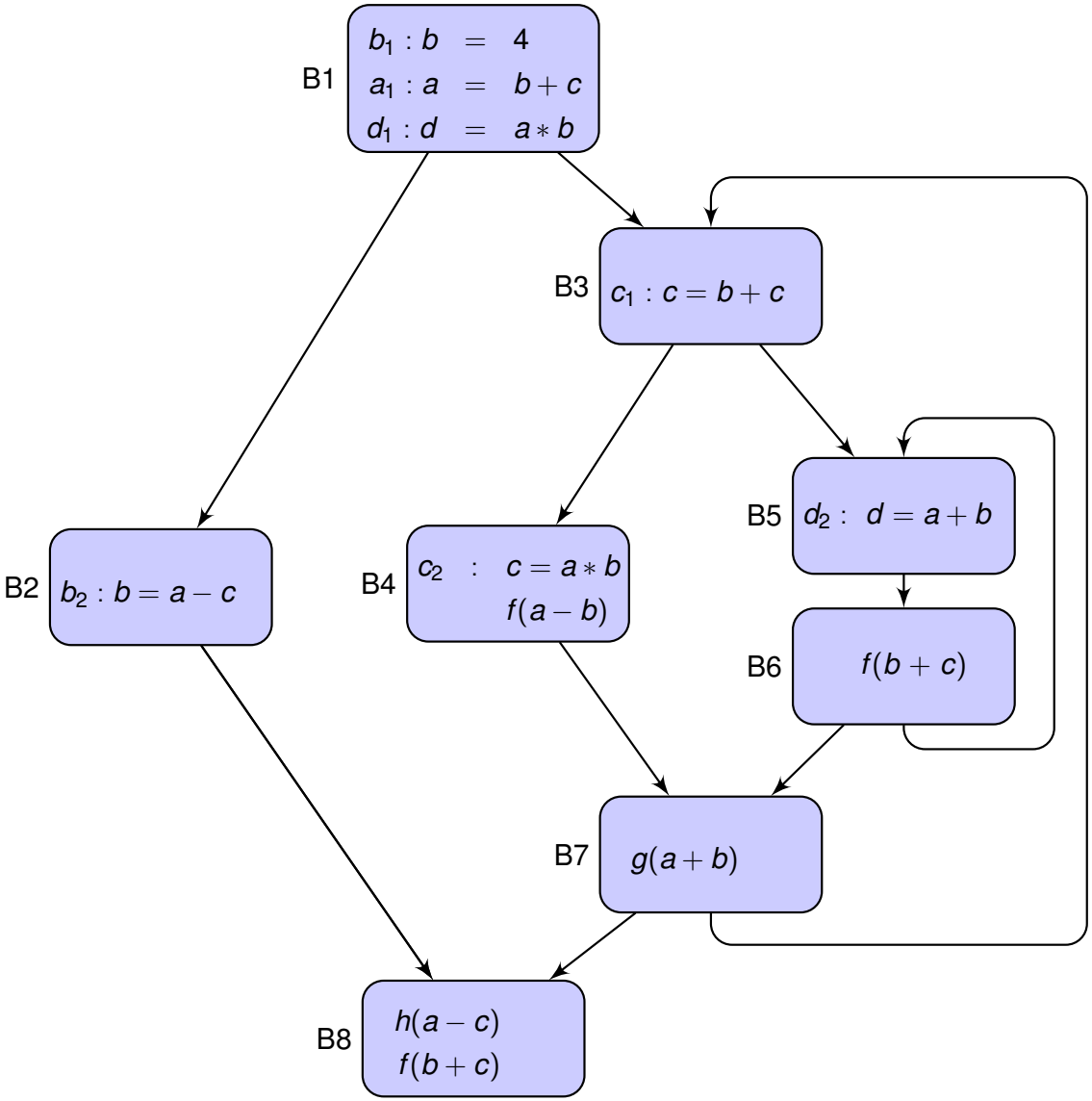
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000		11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B5, B7, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

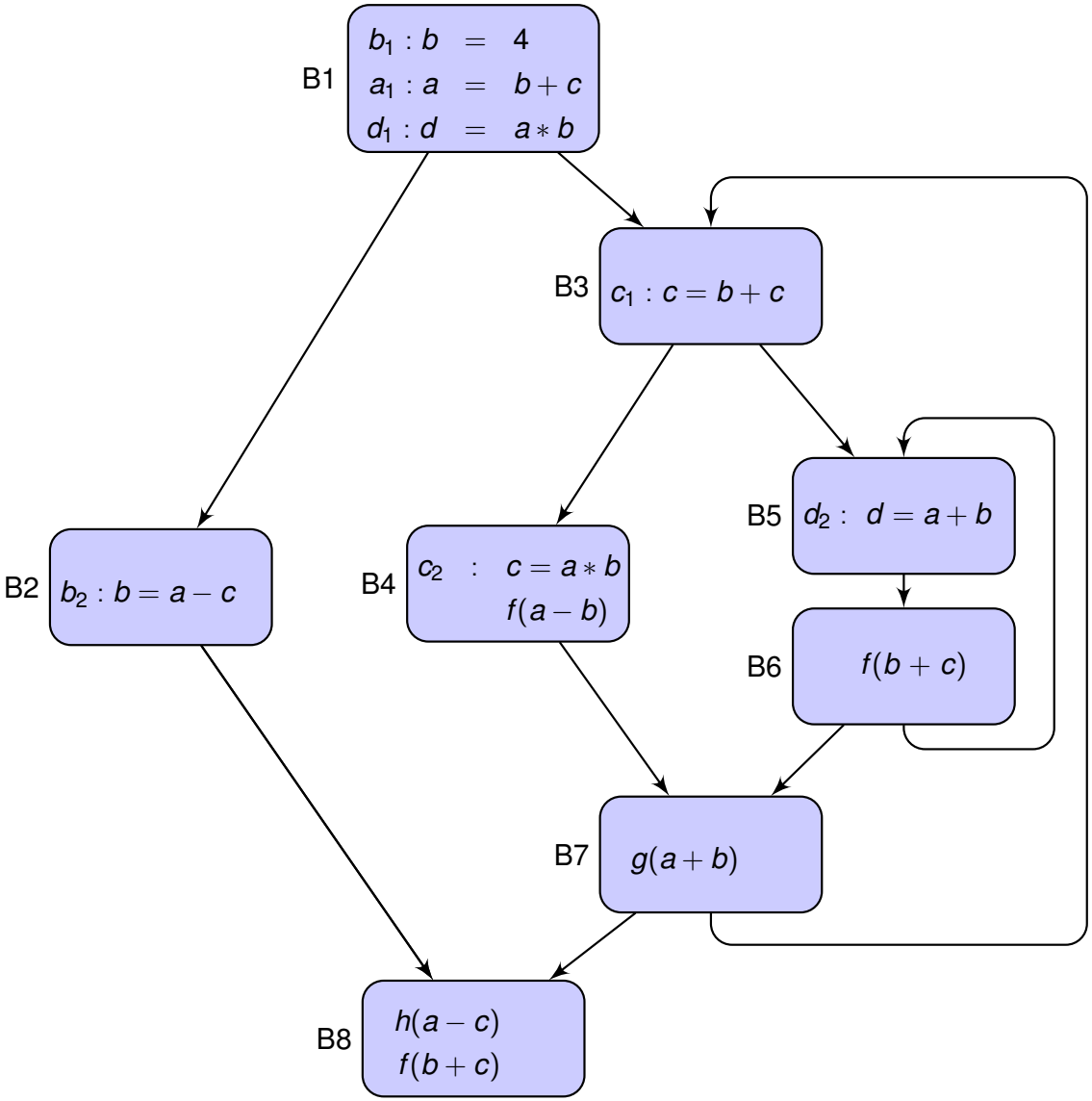
bit pozíció:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11111
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B5, B7, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

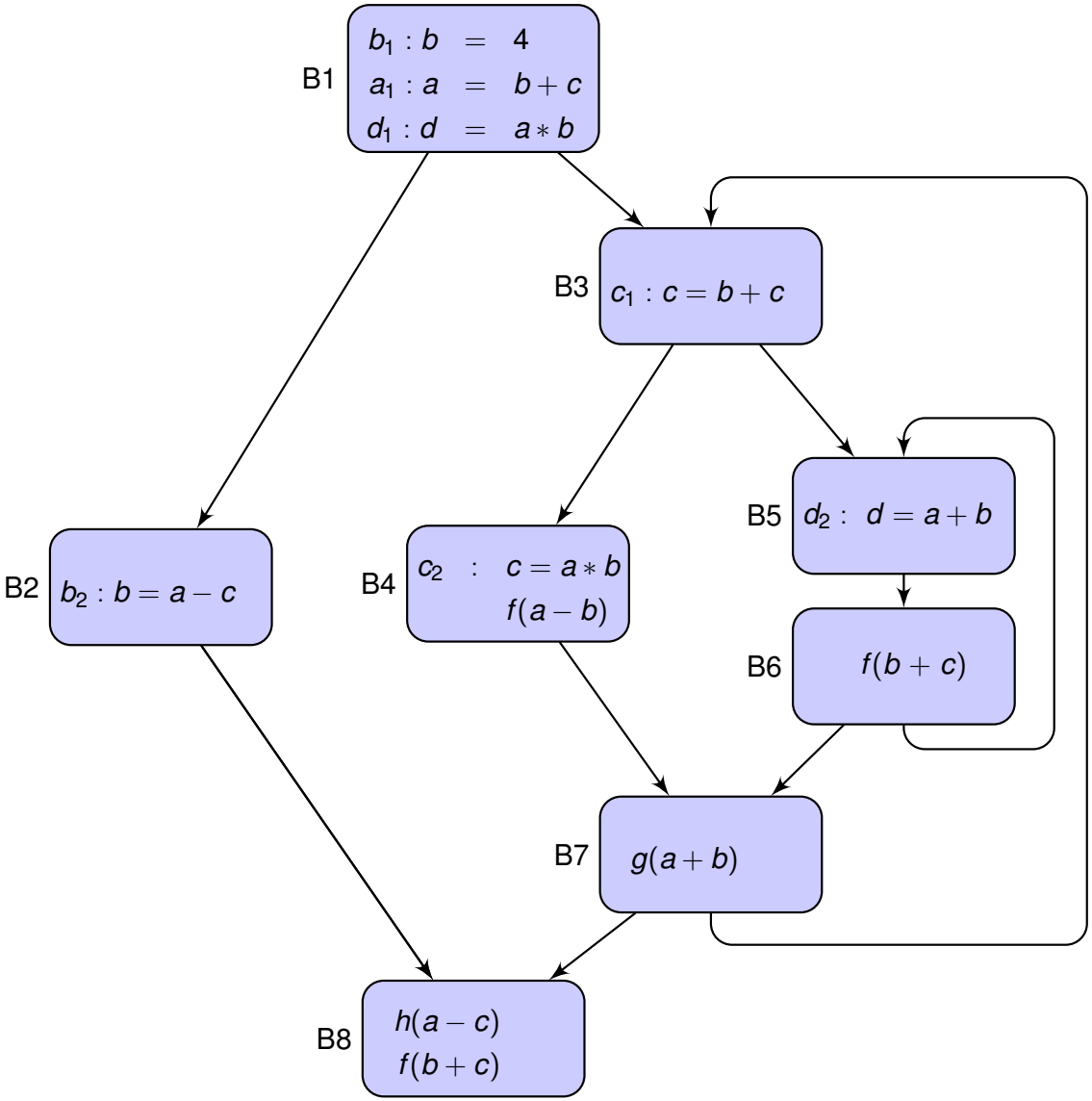
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001		11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B6, B7, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

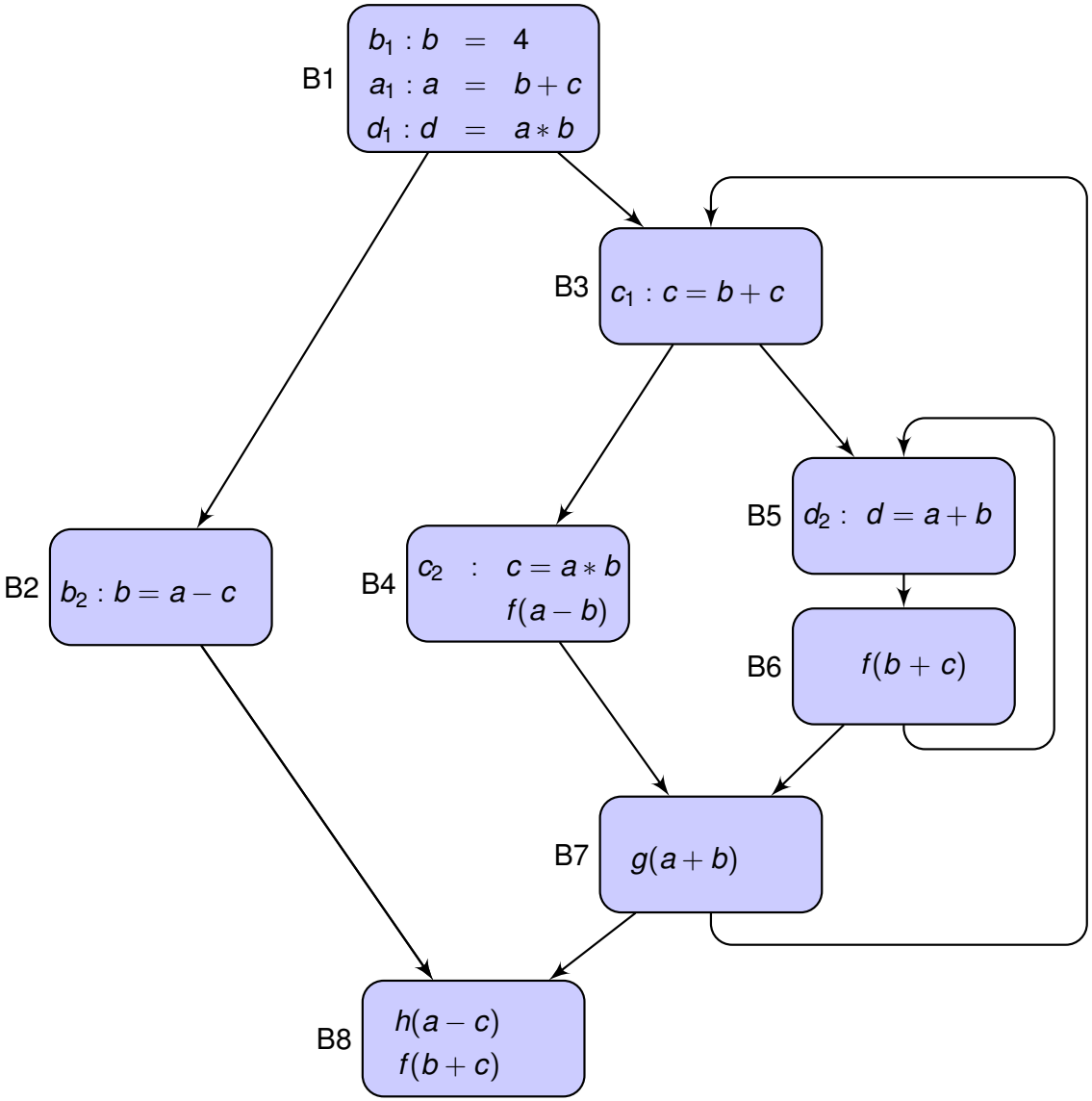
bit pozíció:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11111
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B6, B7, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

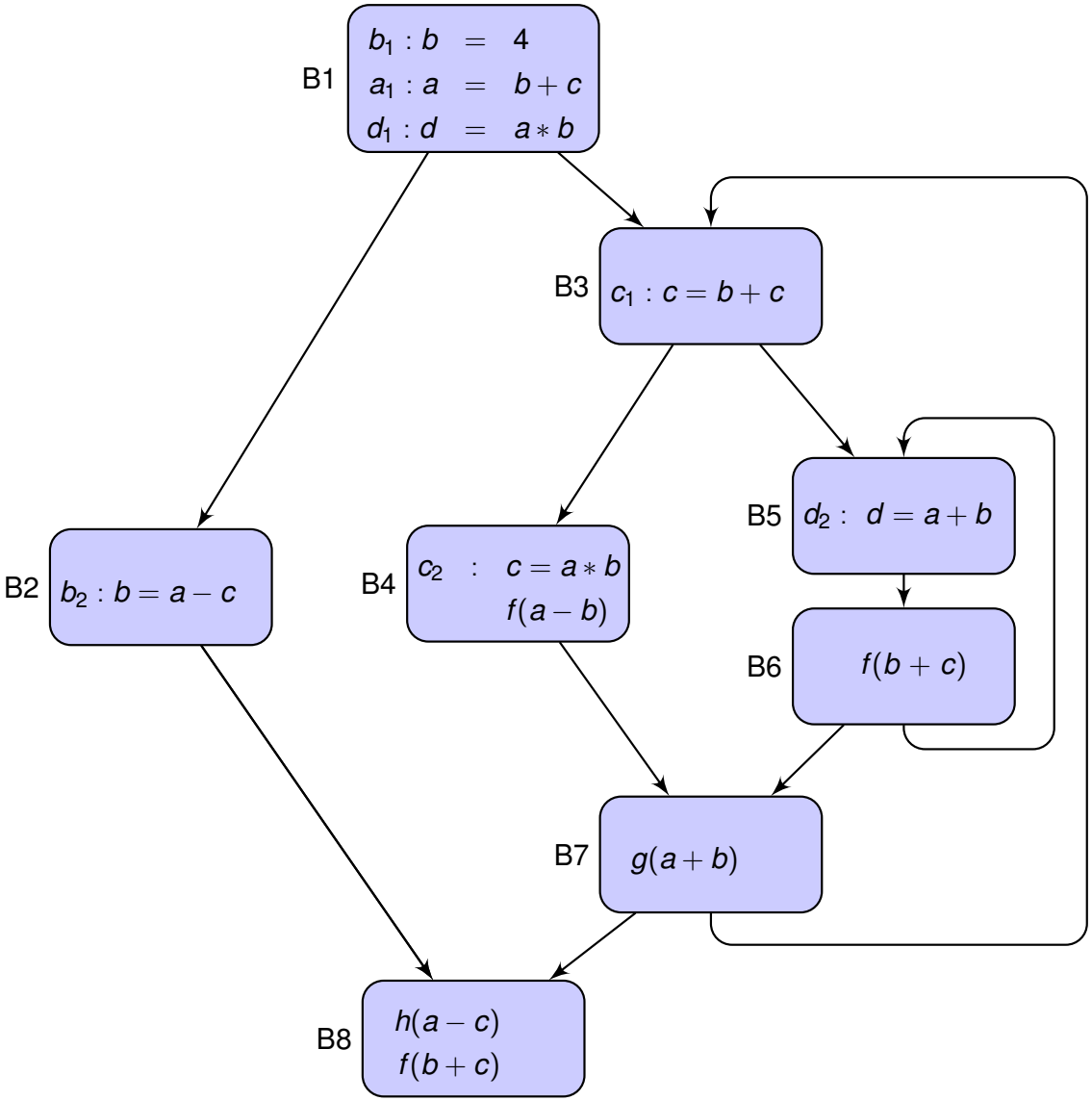
bit pozíció:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B5, B7, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

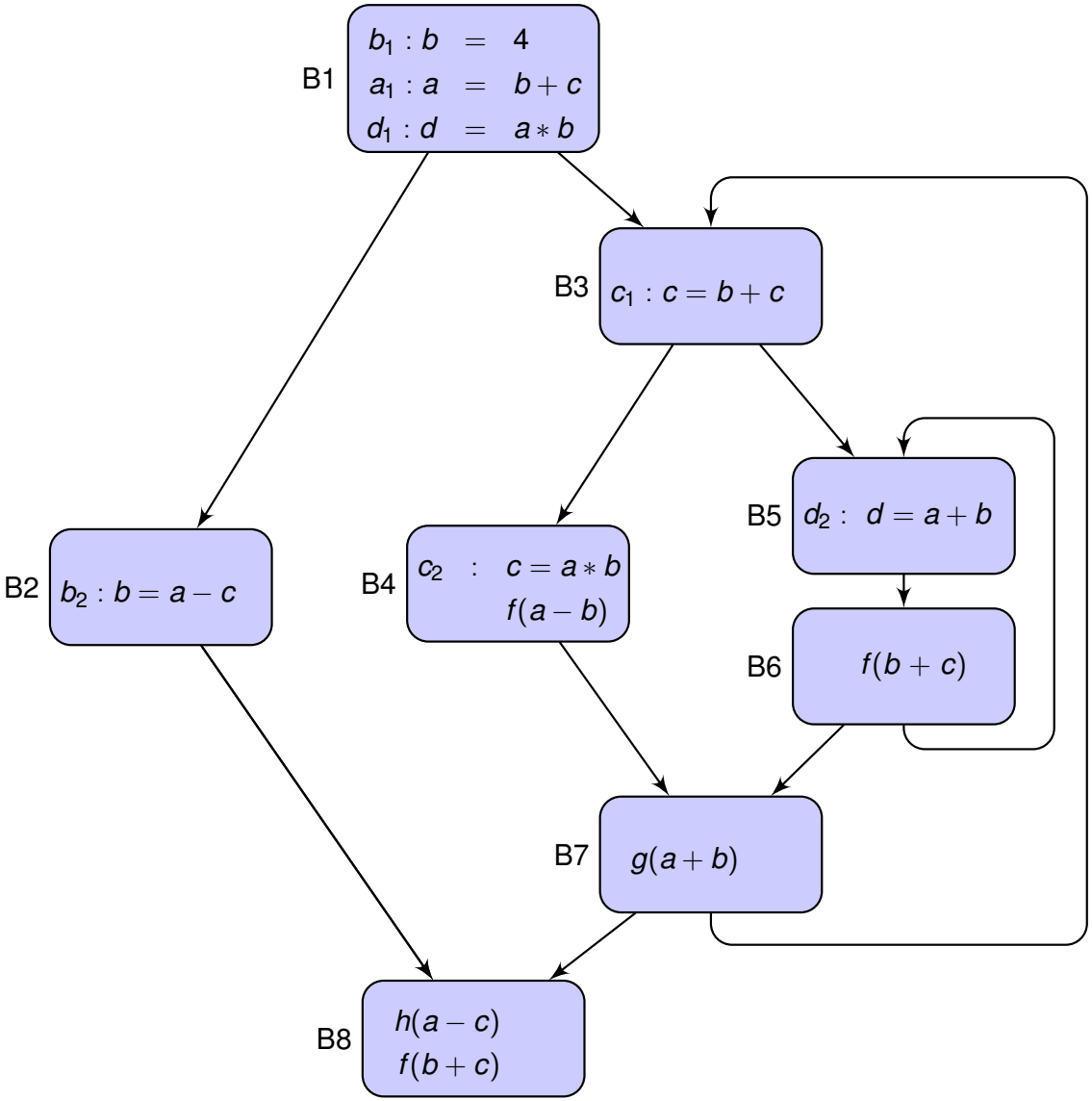
bit pozíció:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	**10001**	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000		11111
B8	00011	00000	00011		11111

TODO: $\{B7, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

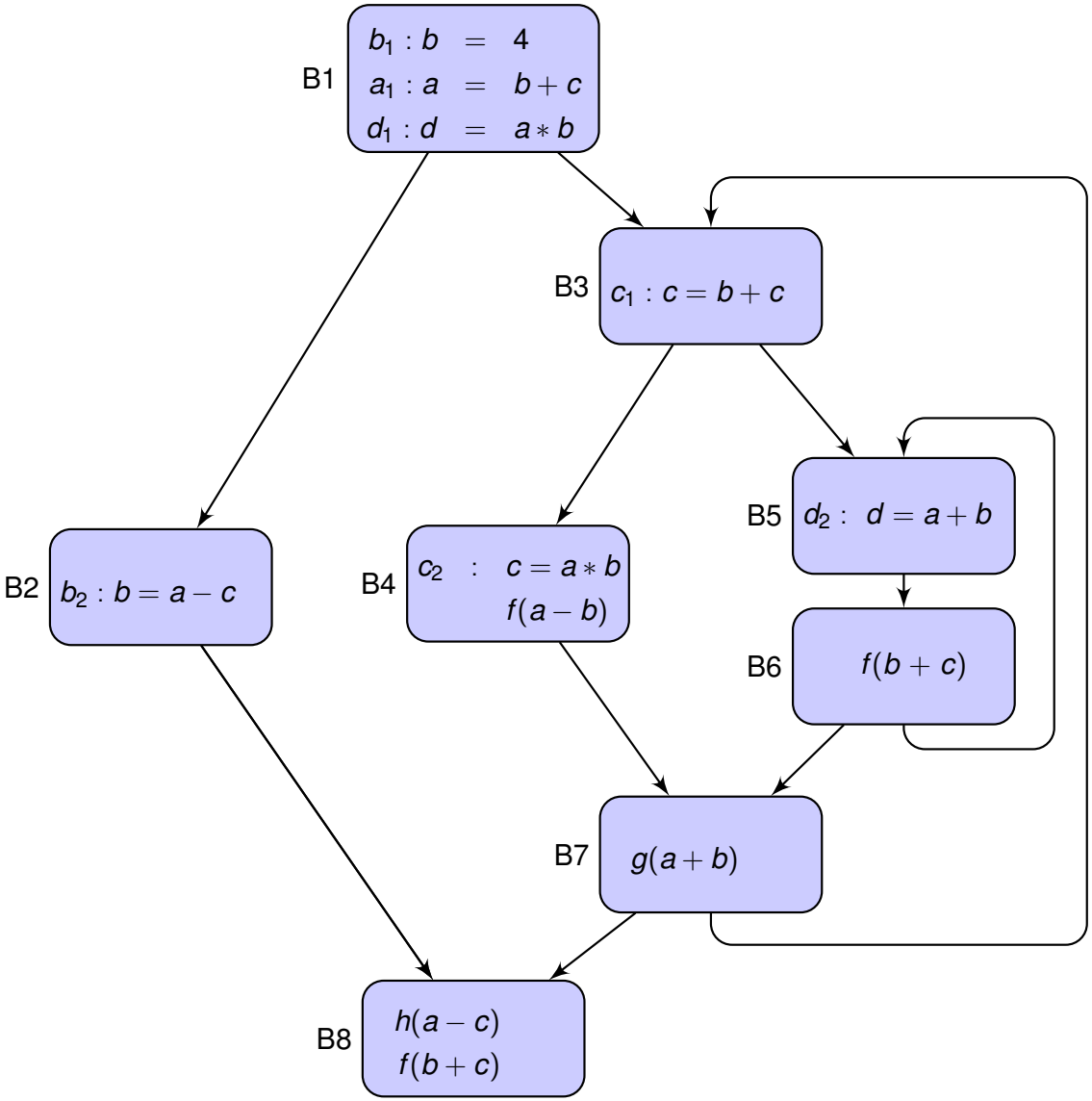
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000	10000	11111
B8	00011	00000	00011		11111

TODO: $\{B7, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

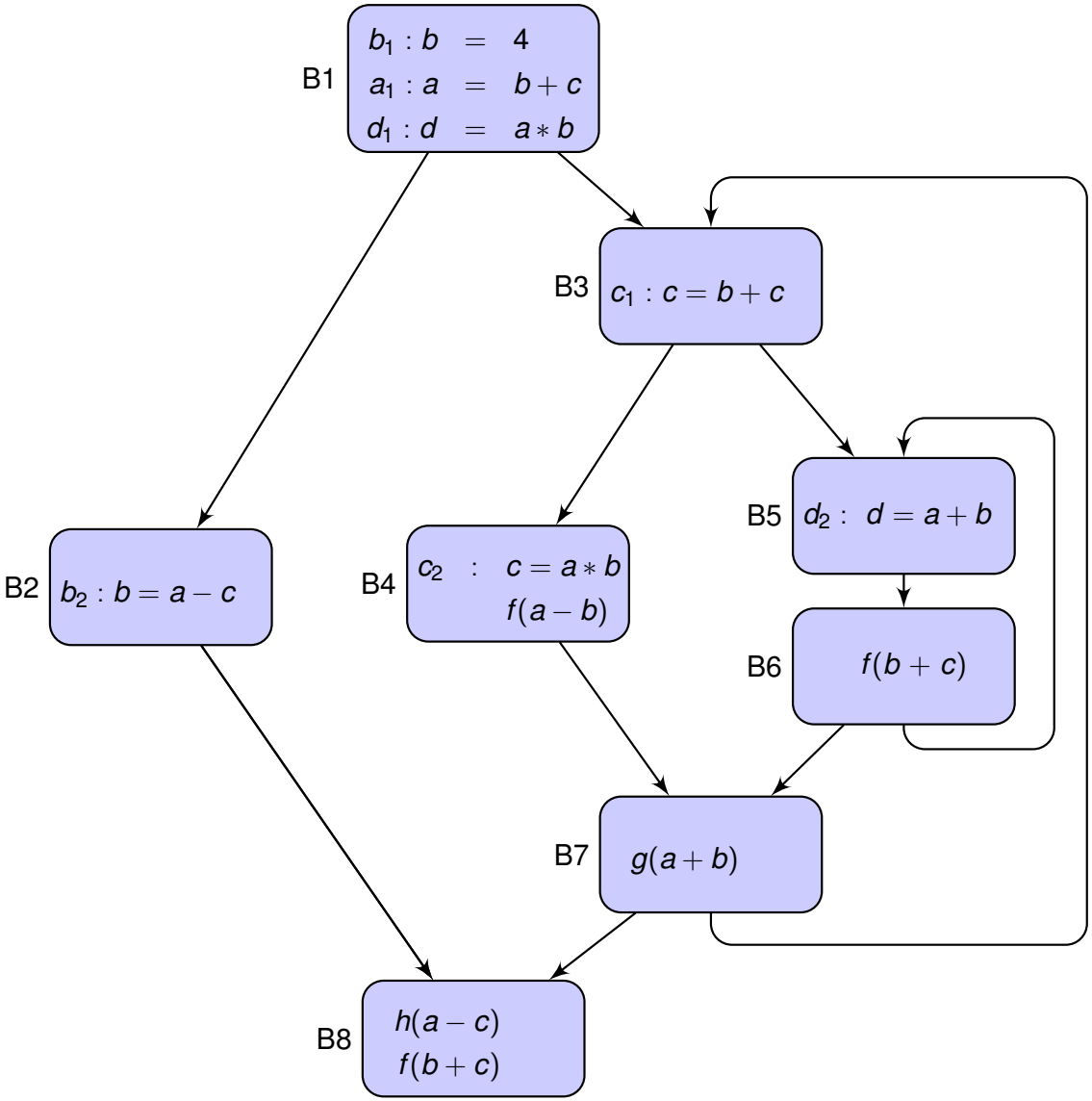
bit pozíció:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10001	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000	10000	11000
B8	00011	00000	00011		11111

TODO: $\{B3, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

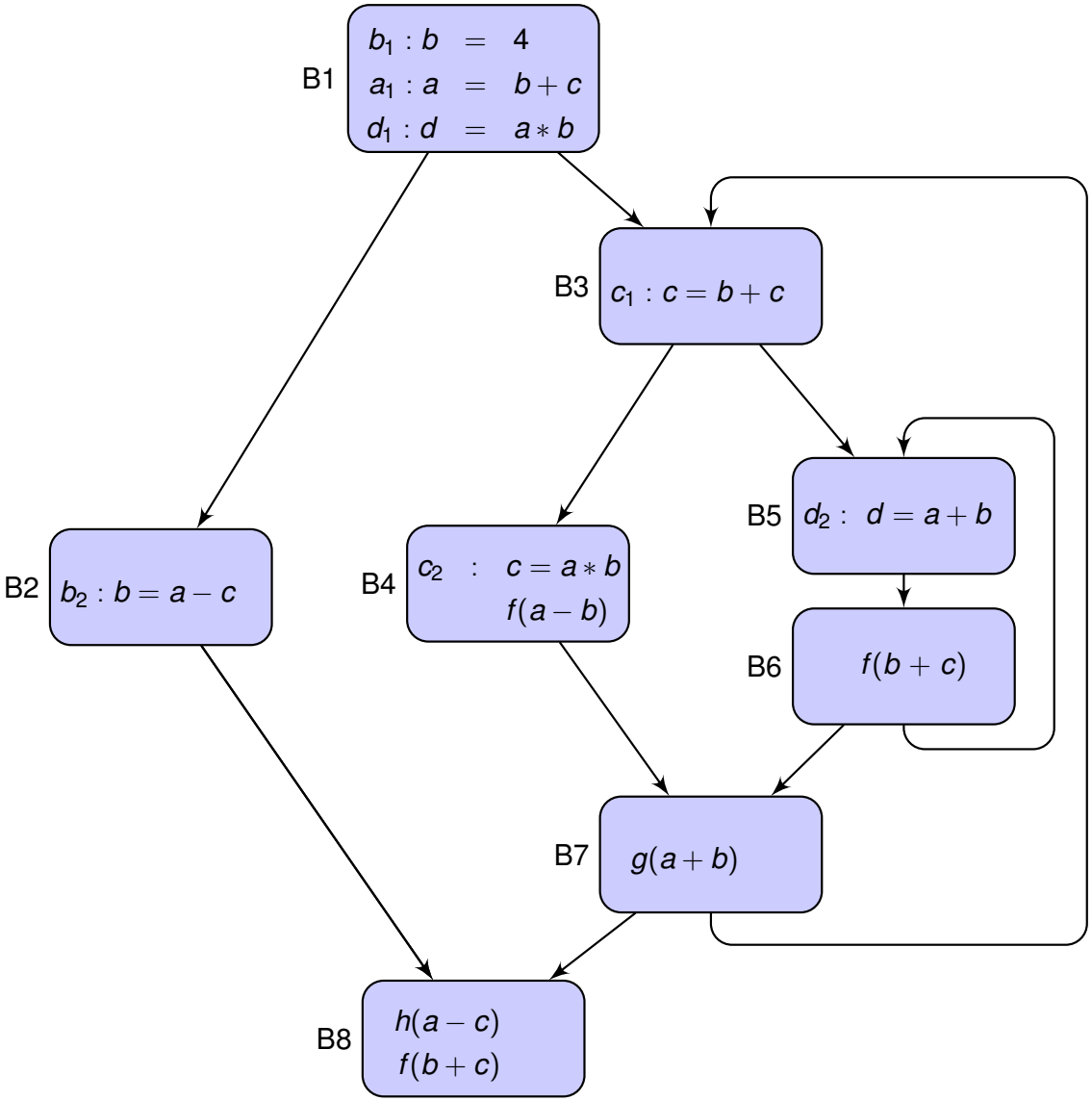
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10000	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000	10000	11000
B8	00011	00000	00011		11111

TODO: $\{B3, B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

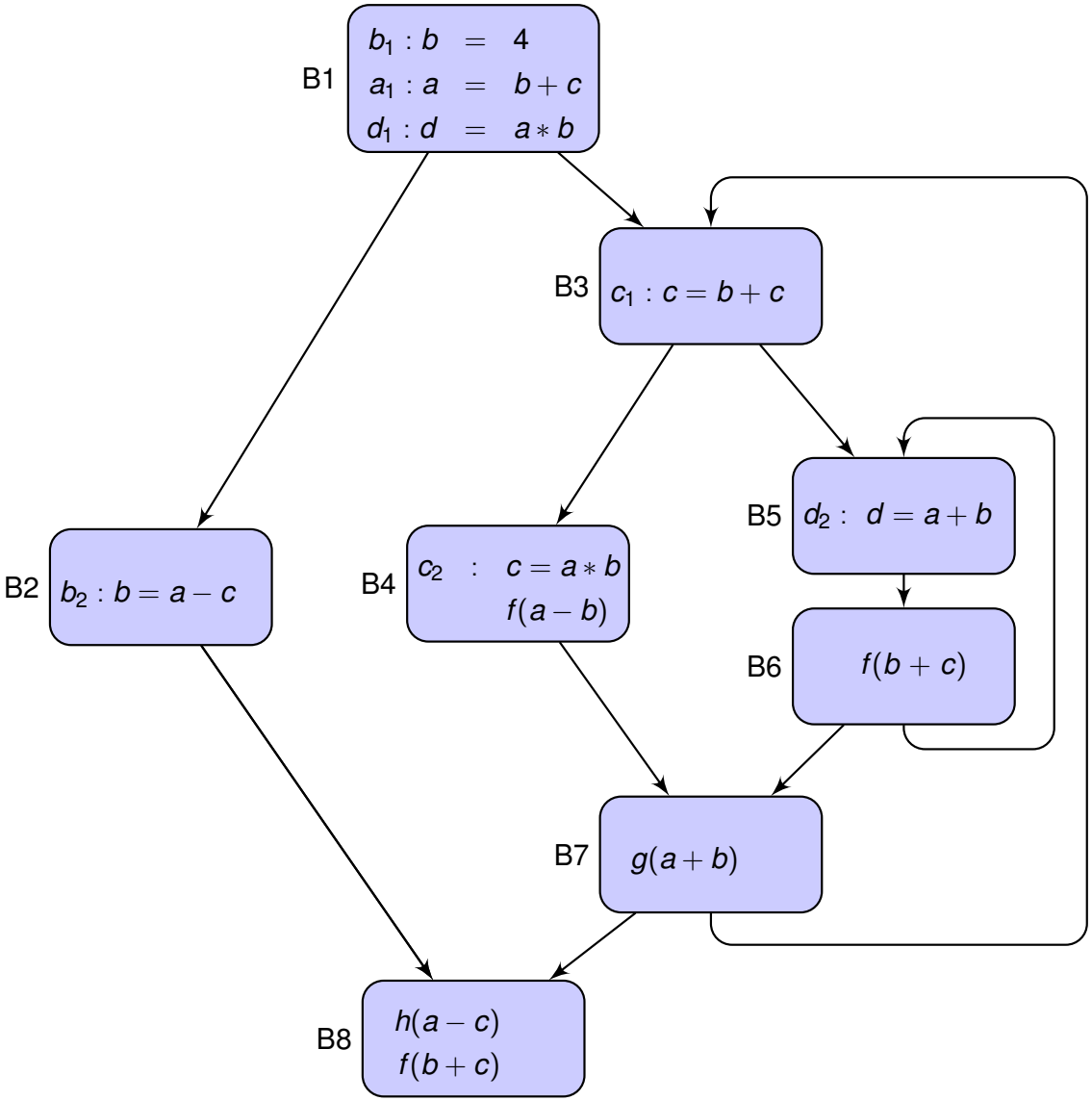
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10000	**10001**
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000	10000	11000
B8	00011	00000	00011		11111

TODO: $\{B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

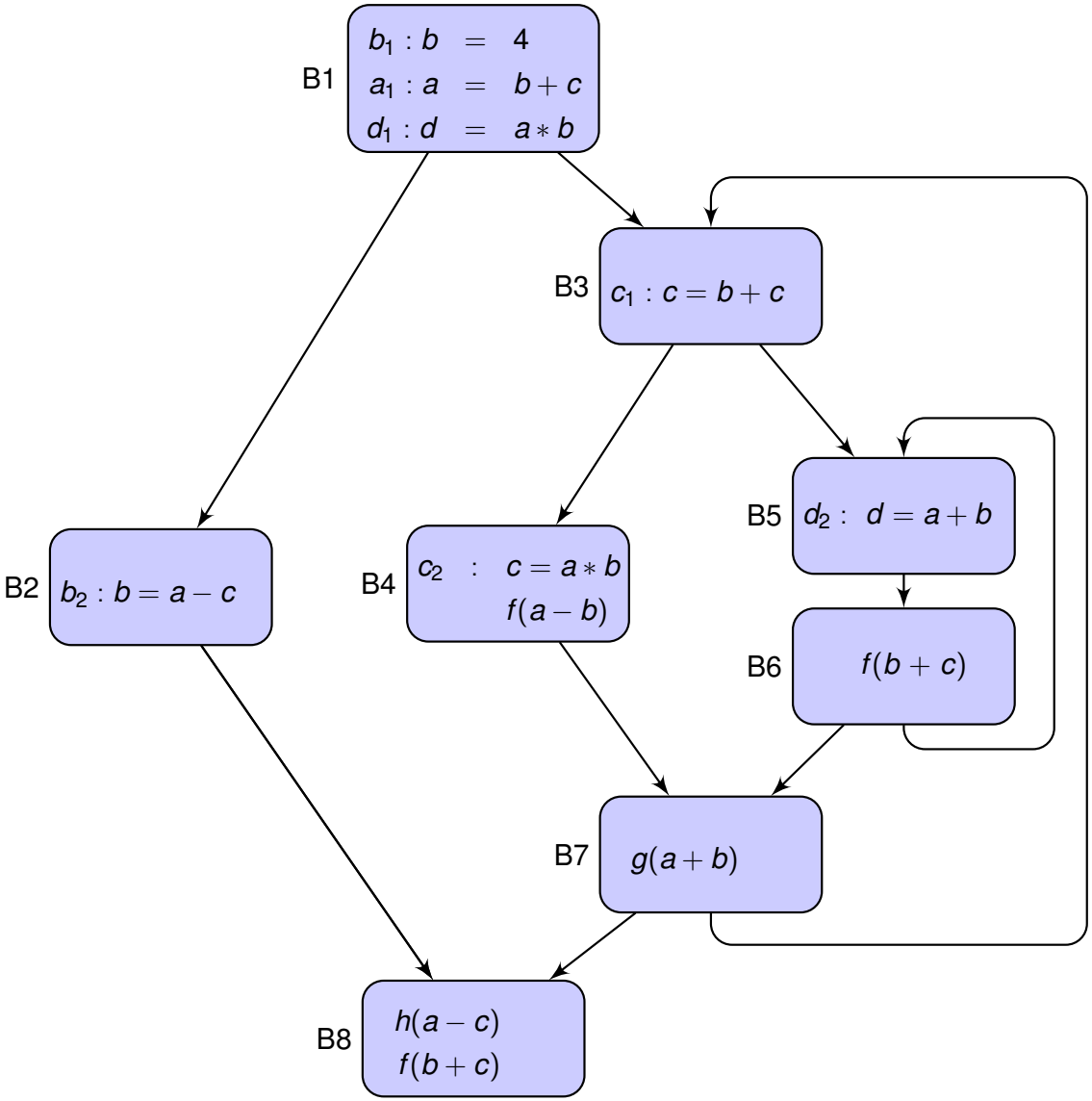
bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10000	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000	10000	11000
B8	00011	00000	00011	00000	11111

TODO: $\{B8\}$

Elérhető kifejezések analízise



$$\mathbb{V}\text{ar} = \{a, b, c, d\}$$
$$\mathbb{D}\text{efs} = a_1, b_1, b_2, c_1, c_2, d_1, d_2$$
$$\mathbb{E}\text{xpr} = \{a * b, a + b, a - b, a - c, b + c\}$$

bit pozició:

kifejezés:	a*b	a+b	a-b	a-c	b+c

Blokk	Lokális információ			Globális információ	
	Gen_n	$Kill_n$	Ant_n	In_n	Out_n
B1	10001	11111	00000	00000	10001
B2	00010	11101	00010	10001	00010
B3	00001	00011	00001	10000	10001
B4	10100	00011	10100	10001	10100
B5	01000	00000	01000	10001	11001
B6	00001	00000	00001	11001	11001
B7	01000	00000	01000	10000	11000
B8	00011	00000	00011	00000	00011

TODO: $Ant_4 \cap In_4 = \{10000\}$ azaz a*b-t felesleges újra kiszámítani.

Lehetséges adatfolyam analízisek

A következő táblázat a lehetséges Gen_n és $Kill_n$ műveleteket jelenti:

Adatok	Műveletek	
Változó $x \in Var$	x értékének olvasása	x változtatása
Kifejezés $e \in Expr$	e kiszámítása	e kifejezés operandusának módosítása
Definiálás $d_i : x = e$ $d_i \in Defs, x \in Var$, $e \in Expr$	d_i előfordulása	x definiálása