AGH

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

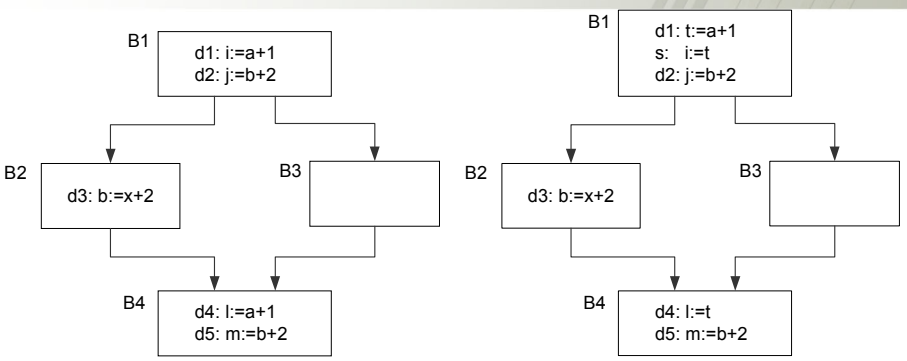
Transformacje optymalizujące

Teoria kompilacji

Dr inż. Janusz Majewski
Katedra Informatyki

AGH

Eliminacja globalnych podwyrażeń



Wyrażenie $a+1$ w definicji d_1 osiąga blok B_4, zatem możemy wyeliminować powtórne obliczanie tego wyrażenia w instrukcji d_4 poprzez podstawienie wartości obliczonej w instrukcji d_1 do zmiennej tymczasowej t i jej wykorzystanie w bloku B_4. Wyrażenie $b+2$ nie osiąga bloku B_4, ponieważ istnieje ścieżka z B_1 poprzez B_2 do B_4, na której mamy redefinicję d_3 argumentu b wyrażenia $b+2$. Zatem to wyrażenie nie jest dostępne w bloku B_4 i nie możemy przeprowadzić operacji analogicznej do poprzedniej.

Algorytm usuwania globalnych wspólnych podwyrażeń

Wejście: Graf przepływu i informacje o dostępnych wyrażeniach

Wyjście: Przeredagowany graf przepływu

Metoda: Dla każdej instrukcji postaci *s: x: = y op z,* gdzie *y op z* jest dostępne na początku bloku zawierającego *s* lub generowane w tym bloku przed *s*, oraz blok ten nie zawiera redefinicji *y* ani *z* przed *s* wykonaj:

1) znajdź takie wyliczenia *w:=y op z*, które osiągają s, przeglądając graf wstecz,
2) utwórz nową zmienną *t*,
3) zamień każdą instrukcję znalezioną w punkcie 1) przez parę
   t:=y op z
   w:=t
4) instrukcję *s x:=y op z* zamień na *x:=t*

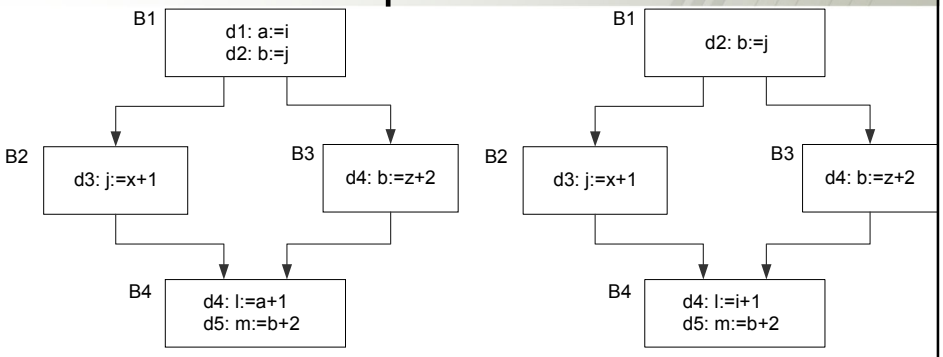
Optymalizacja ta może prowadzić do nadmiernego powiększenia ilości zmiennych tymczasowych, dlatego powinna współpracować z algorytmem eliminacji przenoszenia kopiowania.

Eliminacja przenoszenia kopiowania

Algorytm eliminuje kopiowania *s: x:=y* przez podstawienie *y* za *x*, w tych użyciach *u* zmiennej *x*, dla których spełnione są następujące warunki:

1) *s* musi być jedyną definicją *x* osiągającą *u* (tzn. lista *UDC* w punkcie poprzedzającym *u* zawiera tylko *s*)
2) na każdej ścieżce z *s: x:=y* do *u* nie ma redefinicji *y* (sprawdzenie tego warunku uzyskamy przez rozwiązanie problemu zasięgu instrukcji kopiowania)

Eliminacja przenoszenia kopiowania



Z dwóch instrukcji kopiowania w bloku B_1 grafu lewego można wyeliminować pierwszą (d_1) i podstawić za a wartość zmiennej i w bloku B_4 (d_5). Z drugą (d_2) nie można tak postąpić z dwóch powodów:

1) na ścieżce z B_1 przez B_2 do B_4 mamy redefinicję zmiennej j (d_3)
2) definicja d_2 zmiennej b (którą chcielibyśmy wyeliminować) nie jest jedyną definicją tej zmiennej osiągającą blok B_4 (osiąga go również definicja d_4 z bloku B_3)

Algorytm eliminacji przenoszenia kopiowania

Wejście: Graf przepływu z informacjami o zasięgu instrukcji kopiowania (zbiory *c_in[B]*, ...) oraz dostępnymi listami *UDC* i *DUC*.

Wyjście: Przeredagowany graf przepływu

Metoda: Dla każdej instrukcji *s: x:=y* wykonaj:

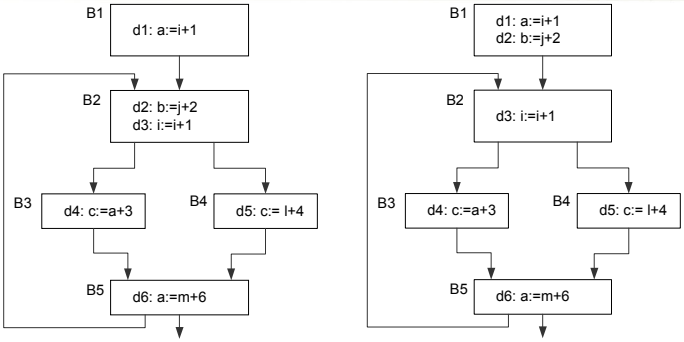
1) określ, które użycia zmiennej x (nazwijmy je u) są osiągalne przez definicję s (lista *DUC*),
2) określ, czy dla każdego użycia u zmiennej x znalezionego w punkcie 1), a znajdującego się w bloku B, s znajduje się w *c_in[B]* (*) lub s znajduje się w bloku B przed u (**),
3) określ, czy w przypadku (*) nie ma wewnątrz bloku B redefinicji zmiennej y przed u (zauważ, że jeśli s jest elementem *c_in[B]*, to s jest jedyną definicją zmiennej x, która osiąga blok B)
4) określ, czy w przypadku (**) nie ma pomiędzy s a u redefinicji zmiennej y (zauważ, że jeżeli s i u są wewnątrz jednego bloku oraz u jest w zasięgu s, to s jest jedyną definicją zmiennej x, która osiąga u)
5) jeżeli warunki 3) lub 4) (odpowiednio dla rozważanego przypadku) są spełnione, to usuń s i zamień wszystkie użycia zmiennej x z punktu 1) na y.

Eliminacja obliczeń niezmienniczych z pętli

Można przenieść przed pętlę takie instrukcje *s: x:=y op z*, które obliczają ciągle tę samą wartość dopóki sterowanie pozostaje w pętli i spełniają poza tym następujące warunki:

1) blok zawierający *s* dominuje nad wszystkimi wyjściami z pętli (czyli nie istnieje ścieżka przez pętlę, która nie przechodziłaby przez ten blok)
2) *s* jest jedyną definicją *x* w tej pętli
3) wszystkie użycia *x* wewnątrz pętli są osiągalne tylko i wyłącznie przez definicję *s*

Eliminacja obliczeń niezmienniczych z pętli – przykład



W pętli z lewego grafu definicje d_2, d_5 i d_6 obliczają w każdym przebiegu pętli tę samą wartość. Do przednagłówka można jednak przesunąć instrukcję d_2. Definicja d_5 nie spełnia warunku 1) – po przesunięciu tej definicji do przednagłówka zmienna *c* przyjmowałaby wartość *l+4* za każdym razem, gdy sterowanie dochodzi do pętli, podczas gdy w rzeczywistości tak być nie musi. Nie jest spełniony również warunek 2). Po przeniesieniu d_5 do przednagłówka po przejściu ścieżki B_2-B_3-B_5-B_2-B_4-B_5 otrzymamy w „ulepszonej" w ten sposób wersji zmienną *c* równą *a+3* podczas gdy w pierwotnej pętli otrzymamy wartość zmiennej *c* równą *l+4*. Definicja d_6 nie spełnia warunku 3). Używana w B_3 zmienna *a* w pierwszym przebiegu pętli ma wartość *i+1*, w każdym następnym *m+5*. Przeniesienie d_6 do przednagłówka zmieniłoby wyraźnie sposób działania programu.

Algorytm wykrywania obliczeń niezmienniczych w pętli

Wejście: pętla *L*, listy UDC
Wyjście: zbiór wyrażeń, które obliczają tę samą wartość dopóki sterowanie pozostaje w pętli

Metoda:

(1) Zamarkuj jako „niezmiennicze" te instrukcje trójadresowe, których wszystkie operandy są bądź stałymi, bądź mają wszystkie swoje definicje poza pętlą *L*.
(2) Powtarzaj krok (3) dopóki występują zmiany.
(3) Zamarkuj jako „niezmiennicze" wszystkie takie instrukcje, które poprzednio nie były zamarkowane, a których wszystkie operandy:
   (a) są stałymi, lub
   (b) mają wszystkie definicje poza pętlą *L*, lub
   (c) mają dokładnie jedną definicję osiągającą rozważaną instrukcję i definicja ta została zamarkowana jako wyrażenie „niezmiennicze" w pętli *L*.

Algorytm eliminacji obliczeń niezmienniczych z pętli

Wejście: pętla *L*, listy UDC, informacje o relacji dominacji w grafie
Wyjście: przeredagowana pętla *L* z przednagłówkiem i (być może) pewnymi instrukcjami przeniesionymi do przednagłówka

Metoda:

(1) Wykonaj poprzedni algorytm wykrywania obliczeń niezmienniczych w pętli *L*.
(2) Dla każdej instrukcji *s* definiującej *x* i zamarkowanej jako „niezmiennicza" sprawdź, czy:
   (a) *s* jest w bloku dominującym wszystkie wyjścia z pętli *L*
   (b) w pętli *L* nie ma definicji *x* innej niż *s*
   (c) wszystkie użycia *x* w *L* są osiągane tylko przez definicję *s*
(3) Przesuń (w porządku wyznaczonym przez poprzedni algorytm) do przednagłówka te instrukcje, które spełniają warunki (2a), (2b) i (2c) pod warunkiem, że jeżeli operandy *s* były definiowane wewnątrz *L*, to definicje te zostały wcześniej przeniesione do przednagłówka.

Algorytm eliminacji obliczeń niezmienniczych z pętli

Uwaga: czasami można osłabić warunek (2a) powyższego algorytmy, formułując go następująco:

(2a) blok zawierający s albo dominuje wszystkie wyjścia z pętli L, albo zmienna x definiowana w s nie jest używana na zewnątrz pętli L.

Jest to przykład tzw. „optymalizacji niebezpiecznej", gdyż czasami przenosi się do przednagłówka wyrażenie, które być może nigdy nie byłoby w pętli wykonane, a po przeniesieniu do przednagłówka będzie wykonane. Na przykład w pętli jest test: „czy y jest różne od 0?" – jeśli tak wykonuje się dzielenie $\frac{x}{y}$. Po ewentualnym wyniesieniu $\frac{x}{y}$ do przednagłówka dzielenie wykona się zawsze, niezależnie od wartości dzielnika, mogąc spowodować błąd wykonania.

Eliminacja zmiennych indukowanych i redukcja mocy

Zmienna x jest nazywana zmienną indukowaną pętli L, jeżeli każdorazowa zmiana jej wartości jest inkrementacją lub dekrementacją o pewną stałą.

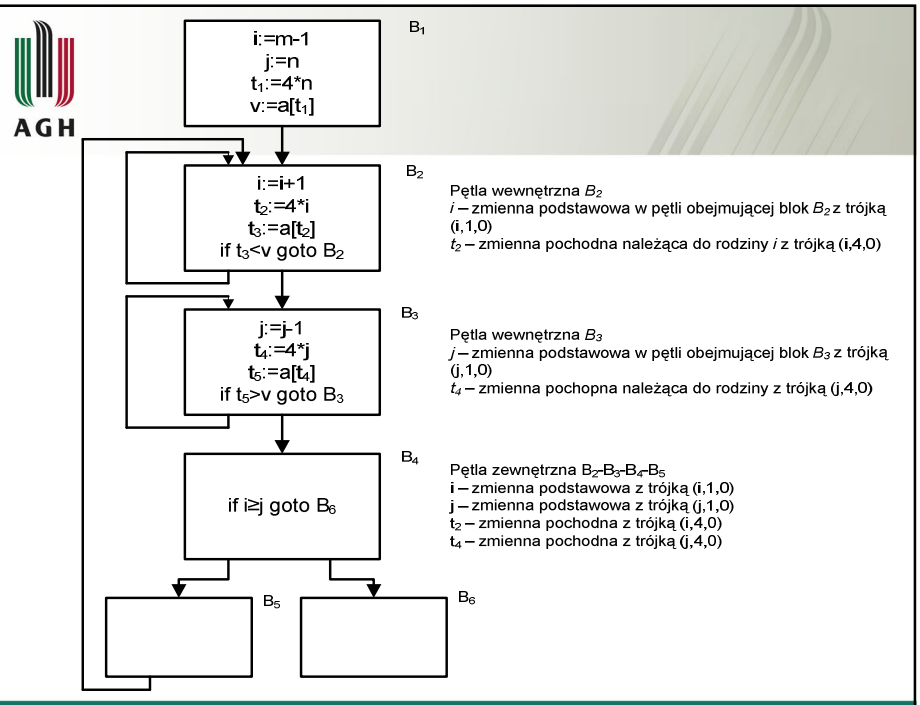
Podstawową zmienną indukowaną jest taka zmienna i, która jest dokładnie jeden raz definiowana wewnątrz pętli L i definicja ta ma postać $i := i + c$ gdzie c jest pewną stałą (np. zmienna sterująca pętli).

Pochodną zmienną indukowaną jest taka zmienna j, która jest dokładnie jeden raz definiowana wewnątrz pętli L, i której wartość jest liniową funkcją pewnej podstawowej zmiennej indukowanej i. Mówimy wtedy, że zmienna j należy do rodziny zmiennej i.

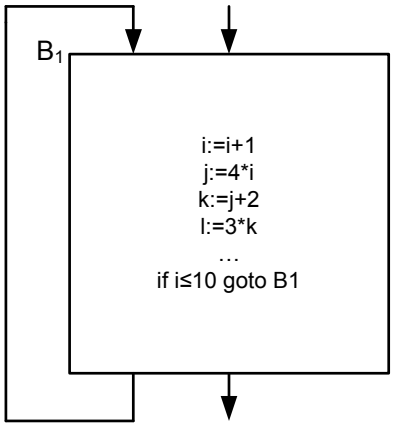
Algorytm wykrywania zmiennych indukowanych (1)

Wejście: pętla L wraz z informacjami o zasięgu definicji i informacjami o wyrażeniach niezmienniczych w pętli L

Wyjście: zbiór zmiennych indukowanych. Z każdą zmienną indukowaną j skojarzona jest trójka (i,c,d), gdzie i jest podstawową zmienną indukowaną, c i d są stałymi, takimi że w miejscu definicji zmiennej j jej wartość wynosi $c \cdot i + d$. Mówimy, że j należy do rodziny i. Podstawowa zmienna indukowana i należy do swojej własnej rodziny.



Przykład



Pętla obejmująca B1:
i – zmienna podstawowa (i,1,0)
j – zmienna pochodna z rodziny *i* z trójką (i,4,0)
k – zmienna pochodna z rodziny *i* z trójką (i,4,2)
[k=j+2=4*i+2]
l – zmienna pochodna z rodziny *i* z trójką (i,12,6)
[l=3*k=3*(4*i+2)=12*i+6]

Algorytm wykrywania zmiennych indukowanych (2)

Metoda:

(1) Znajdź wszystkie podstawowe zmienne indukowane *i* przeglądając instrukcje pętli *L*. Zmienna *i* musi mieć dokładnie jedną definicję w pętli *L* w postaci $i := i + a$, gdzie *a* jest stałą. Z każdą podstawową zmienną indukowaną *i* skojarz trójkę $(i,1,0)$.

(2) Poszukaj wszystkich zmiennych *k* mających dokładnie jedną definicję w pętli *L* w jednej z poniższych postaci:

$k := j \cdot b, \quad k := b \cdot j, \quad k := \frac{j}{b}, \quad k := j \pm b, \quad k := b \pm j$

gdzie: *b* – stała, *j* – zmienna indukowana, podstawowa lub pochodna.
Jeśli *j* jest podstawową zmienną indukowaną, wówczas *k* jest pochodną zmienną indukowaną i należy do rodziny *j*. Trójka dla *k* zależy od postaci instrukcji definiującej tę zmienną. Na przykład jeśli *k* definiowane jest jako $k := j \cdot b$, to trójka dla *k* ma postać $(j,b,0)$; jeżeli $k := \frac{j}{b}$, to trójka dla *k* ma postać $(j,\frac{1}{b},0)$; jeżeli $k := j + b$, wówczas trójka dla *k* to $(j,1,b)$.

Algorytm wykrywania zmiennych indukowanych (3)

Jeśli j nie jest podstawową zmienną indukowaną, to wtedy j jest zmienną pochodną i należy do rodziny zmiennej podstawowej i. Wówczas należy sprawdzić dodatkowe wymagania:

(a) nie ma być podstawienia dla zmiennej i pomiędzy jedyną definicją j w L, a jedyną definicją k w L.

(b) żądna definicja j spoza pętli L nie osiąga definicji k.

Gdy wymagania te są spełnione, k jest zmienną pochodną należącą do rodziny i.

Wtedy tworzymy trójkę dla zmiennej k na podstawie trójki (i,c,d) dla zmiennej j i na podstawie instrukcji definiującej zmienną k. Na przykład definicja $k := b \cdot j$ prowadzi do trójki $(i, b \cdot c, b \cdot d)$ dla zmiennej k.

Algorytm redukcji mocy wykorzystujący zmienne indukowane

Wejście: pętla L wraz z informacją o zasięgu definicji oraz informacje o rodzinach zmiennych indukowanych uzyskane w wyniku wykonania poprzedniego algorytmu

Wyjście: przeredagowana pętla

Metoda:

Rozważamy kolejno każdą podstawową zmienną indukowaną i. Dla każdej pochodnej zmiennej indukowanej j z rodziny zmiennej i (zmienna j scharakteryzowana trójką (i,c,d)) wykonaj:

(1) Utwórz nową zmienną s (jeśli dwie zmienne pochodne j_1 i j_2 mają takie same trójki, utwórz dla obu tylko jedną nową zmienną).

(2) Zamień podstawienie do zmiennej j na $j := s$.

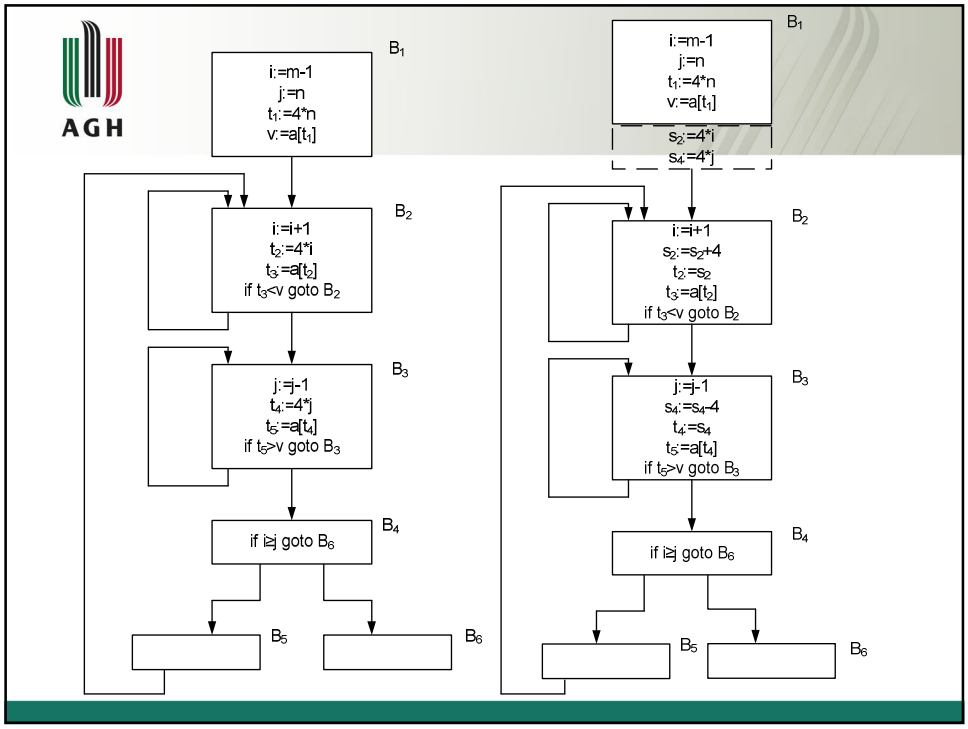
(3) Bezpośrednio po podstawieniu $i := i + a$ (gdzie: a – stała) dodaj $s := s + c \cdot a$, gdzie wyrażenie $c \cdot a$ jest obliczone i podane jako stała, gdyż c oraz a są stałymi. Umieść zmienną s w rodzinie zmiennej podstawowej i z odpowiadającą jej trójką (i,c,d).

(4) Wstaw na koniec przednagłówka pętli

$s := c \cdot i$ ($s := i$ gdy $c = 1$)

$s := s + d$ (opuść, gdy $d = 0$)

Zauważ, że s jest także pochodną zmienną indukowaną z rodziny zmiennej i.



Algorytm eliminacji zmiennych indukowanych (1)

Wejście: pętla L wraz z informacją o zasięgu definicji, o wyrażeniach niezmienniczych i o życiu zmiennych.

Wyjście: przeredagowana pętla.

Metoda:

(1) Rozważamy kolejno każdą podstawową zmienną indukowaną, która używana jest wyłącznie do obliczania innych zmiennych indukowanych ze swojej rodziny oraz w wyrażeniach w skokach warunkowych wewnątrz pętli. Weź pewną zmienną j z rodziny zmiennej i, najlepiej taką, aby c i d z jej trójki były możliwie najprostsze (na przykład $c = 1$, $d = 0$). Zmodyfikuj każde wyrażenie w skokach warunkowych tak, aby zamiast i było tam używane j. Załóżmy dalej, że $c > 0$.

Skok warunkowy **`if i rel_op x goto B`**, gdzie x nie jest zmienną indukowaną, jest zamieniany na:

`r:=c*x` ($r := x$, gdy $c = 1$)

`r:=r+d` (opuść, gdy $d = 0$)

`if j rel_op r goto B`

gdzie r jest nową zmienną tymczasową.

Algorytm eliminacji zmiennych indukowanych (2)

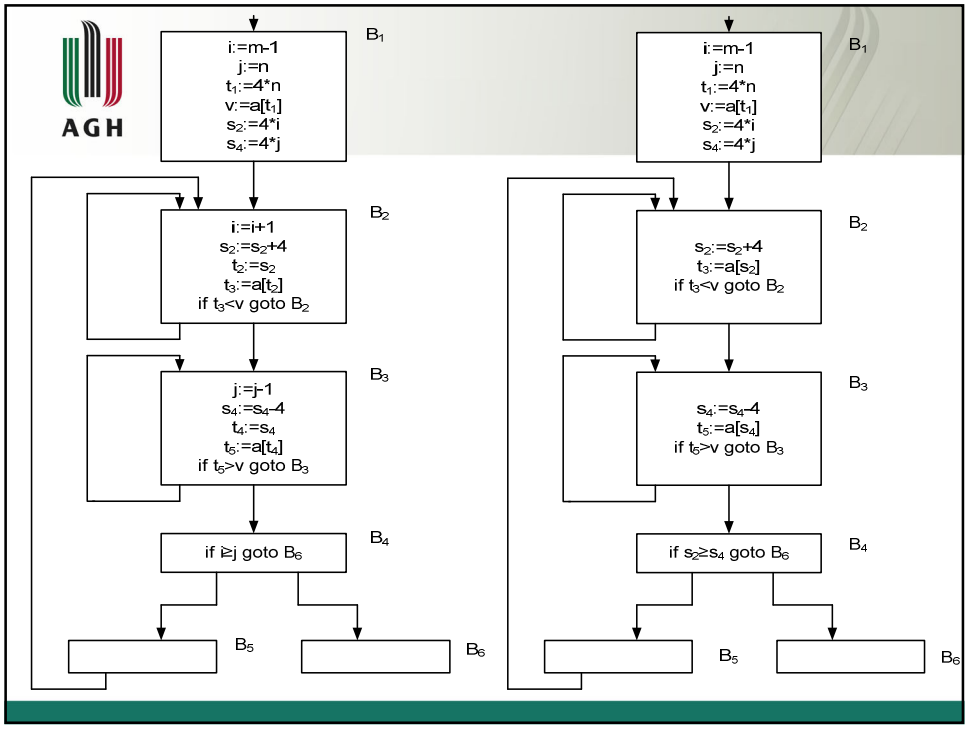
Skok warunkowy
`if i rel_op a goto B`, gdzie *a* jest stałą jest zamieniany na:
`if j rel_op c*a+d goto B`, gdzie wyrażenie $c \cdot a + d$ jest wyliczone. W przypadku, kiedy w teście skoku warunkowego są używane dwie zmienne indukowane i_1 i i_2 (**`if i1 rel_op i2 goto B`**), sprawdzamy, czy obie mogą być usunięte. W najprostszym przypadku, gdy j_1 ma trójkę (i_1, c_1, d_1), j_2 zaś (i_2, c_2, d_2) oraz $c_1 = c_2$ i $d_1 = d_2$, wtedy **`i1 rel_op i2`** jest równoważne **`j1 rel_op j2`**.

W bardziej złożonych przypadkach modyfikacja testu przy skokach warunkowych może nie być warta zachodu, gdyż musielibyśmy (w ogólnym przypadku) wprowadzić dwa dodatkowe mnożenia i jedno dodawanie.

Po przekształceniu skoków warunkowych usuwamy wszystkie podstawienia do wyeliminowanych podstawowych zmiennych indukowanych, gdyż zmienne te są teraz bezużyteczne.

Algorytm eliminacji zmiennych indukowanych (3)

(2) Rozważamy teraz każdą zmienną indukowaną *j*, dla której podstawienie $j := s$ zostało wstawione podczas wykonania poprzedniego algorytmu. Sprawdzamy, czy nie ma jakiegoś podstawienia do zmiennej *s* pomiędzy instrukcją $j := s$ i jakimkolwiek użyciem zmiennej *j*. Jeśli nie ma, zastępujemy wszystkie użycia *j* przez *s* i usuwamy podstawienie $j := s$.



Zmienne indukowane a wyrażenia niezmiennicze w pętli

W algorytmach wykrywania zmiennych indukowanych i redukcji mocy można dopuścić wyrażenia niezmiennicze oprócz stałych. Jednak wówczas określenie (i,c,d) dla zmiennej indukowanej j wymagałoby znajomości wartości wyrażenia niezmienniczego. Obliczenie trójki musiałoby się odbyć w przednagłówku, w pewnych przypadkach wymagałoby kilku instrukcji trójadresowych. Algorytm eliminacji zmiennych indukowanych wymaga z kolei znajomości znaku c; gdyby c było ujemne, należałoby zmienić operator relacji w teście skoku warunkowego. Dlatego też z reguły ograniczamy się tylko do stałych przy wykrywaniu zmiennych indukowanych.