Weryfikacja Oprogramowania

semestr zimowy 2014/2015

Grzegorz Herman

Informatyka Analityczna
tcs@jagiellonian

Warunki zaliczenia

Punkty

- 5 zadań na Satori po 4 punkty każde
- egzamin: 3 z 4 pytań, po 6 punktów każde

Ocena

- z ćwiczeń: Satori
- końcowa: wszystko

Progi

$\leqslant$50%	2.0
50–60%	3.0
60–70%	3.5
70–80%	4.0
80–90%	4.5
$>$90%	5.0

Bonus

- implementacja metody z wykładu – podwyższenie oceny

Plan ćwiczeń

C/C++

- podstawowe narzędzia
- testy w modelu „black-box”
- analiza programów wielowątkowych

Java

- unit testing
- mock objects
- pokrycie kodu, mutation testing
- język specyfikacji JML
- instrumentacja

Testowanie UI

- web-based UI
- desktop UI

Plan wykładów

Wstęp

- grafowe reprezentacje programów

Część 1: analiza dynamiczna

- techniki instrumentacji
- wykrywanie data races
- analiza wpływu

Część 2: jakość i generowanie testów

- mutation testing
- testy pokrywające ścieżkę/punkt
- generowanie testów strukturalnych

Plan wykładów (cd.)

Część 3: analiza statyczna

- wnioskowanie oparte o type inference
- analiza wskazywania
- przekroje

Część 4: model checking

- logika Hoare'a
- logiki temporalne
- algorytmy model checking

$\Rightarrow$

- wskazówki
- optymalizacje

Analiza statyczna

- nie uruchamia programu
- ogólne własności programu
- więcej informacji
- trudniejsza

Analiza dynamiczna

- uruchamia program
- konkretny przebieg programu
- mniej informacji
- (względnie) prostsza

$\Leftarrow$

- feedback

$\Rightarrow$

- wskazówki
- optymalizacje

Analiza statyczna

- nie uruchamia programu
- ogólne własności programu
- więcej informacji
- trudniejsza

Analiza dynamiczna

- uruchamia program
- konkretny przebieg programu
- mniej informacji
- (względnie) prostsza

$\Leftarrow$

- feedback

Wejście: kod źródłowy

Założenia

- język imperatywny
- pojedynczy wątek
- determinizm
- pojedyncza funkcja (analiza *intra*proceduralna)

Uproszczenia wstępne

- dekonstrukcja struktur wysokiego poziomu
- przepływ sterowania zamieniony na skoki warunkowe

```
int ten() {
  int i;
  for (i=0; i<10; ++i);
  return i;
}
```

⇒

```
  i = 0;
checkfor:
  if (i>=10) goto endfor;
  ++i;
  goto checkfor;
endfor:
  return i;
```

Wejście: kod źródłowy

Założenia

- język imperatywny
- pojedynczy wątek
- determinizm
- pojedyncza funkcja (analiza *intra*proceduralna)

Uproszczenia wstępne

- dekonstrukcja struktur wysokiego poziomu
- przepływ sterowania zamieniony na skoki warunkowe

```
int ten() {
  int i;
  for (i=0; i<10; ++i);
  return i;
}
```

$\Rightarrow$

```
  i = 0;
checkfor:
  if (i>=10) goto endfor;
  ++i;
  goto checkfor;
endfor:
  return i;
```

Control Flow Graph

Basic block

- liniowy ciąg instrukcji
- skoki „na zewnątrz” tylko z ostatniej
- skoki „do wewnątrz” tylko do pierwszej

Control Flow Graph $G = (V, E, s, t)$

- V – zbiór basic blocks
- $E \subseteq V \times V$ – możliwy przepływ sterowania
- $s \in V$ – instrukcja wejściowa
- $t \in V$ – instrukcja wyjściowa

Regularyzacja

- każdy $v \in V$ osiągalny z s
- t osiągalny z każdego $v \in V$

Control Flow Graph

Basic block

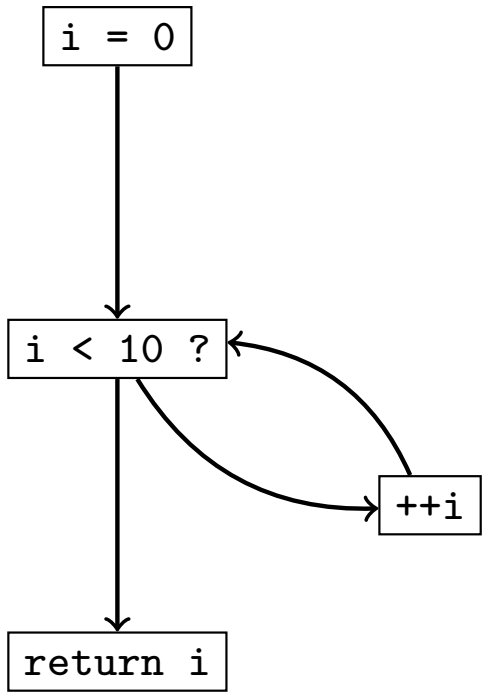
- liniowy ciąg instrukcji
- skoki „na zewnątrz” tylko z ostatniej
- skoki „do wewnątrz” tylko do pierwszej

Control Flow Graph $G = (V, E, s, t)$

- V – zbiór basic blocks
- $E \subseteq V \times V$ – możliwy przepływ sterowania
- $s \in V$ – instrukcja wejściowa
- $t \in V$ – instrukcja wyjściowa

Regularyzacja

- każdy $v \in V$ osiągalny z s
- t osiągalny z każdego $v \in V$



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

bezpośrednie dominatory

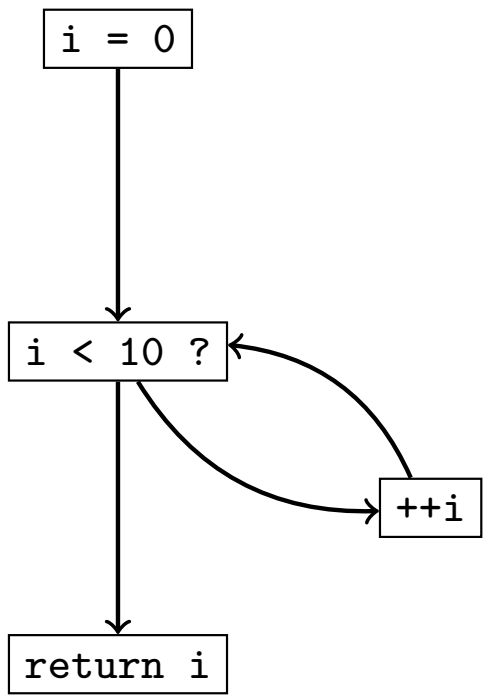
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

postdominacja

to dominacja od t po odwróconych krawędziach



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

Bezpośrednie dominatory

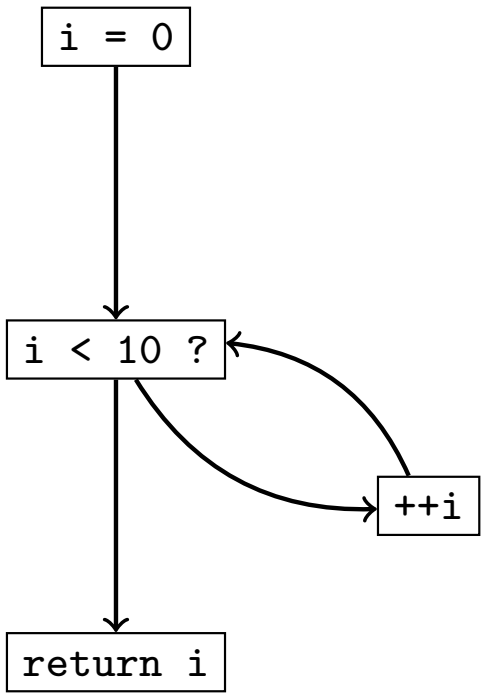
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

Postdominacja

to dominacja od t po odwróconych krawędziach



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

Bezpośrednie dominatory

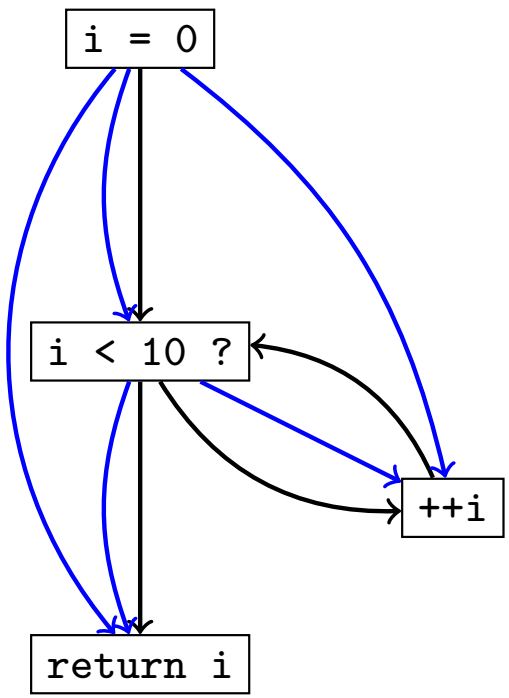
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

Postdominacja

to dominacja od t po odwróconych krawędziach



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

Bezpośrednie dominatory

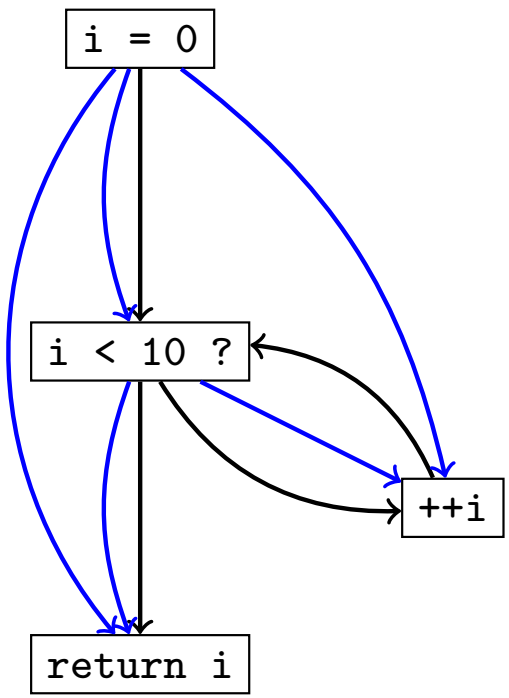
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

Postdominacja

to dominacja od t po odwróconych krawędziach



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

Bezpośrednie dominatory

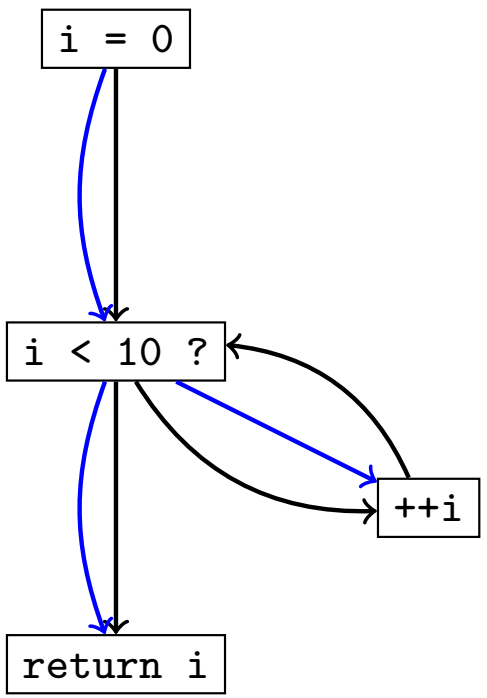
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

Postdominacja

to dominacja od t po odwróconych krawędziach



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

Bezpośrednie dominatory

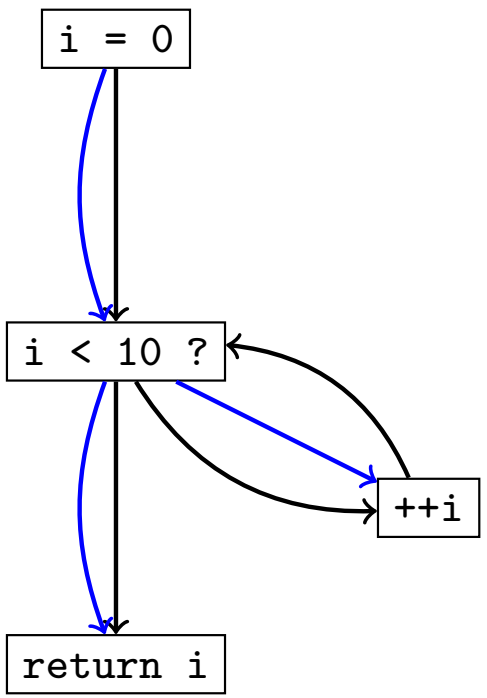
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

Postdominacja

to dominacja od t po odwróconych krawędziach



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

bezpośrednie dominatory

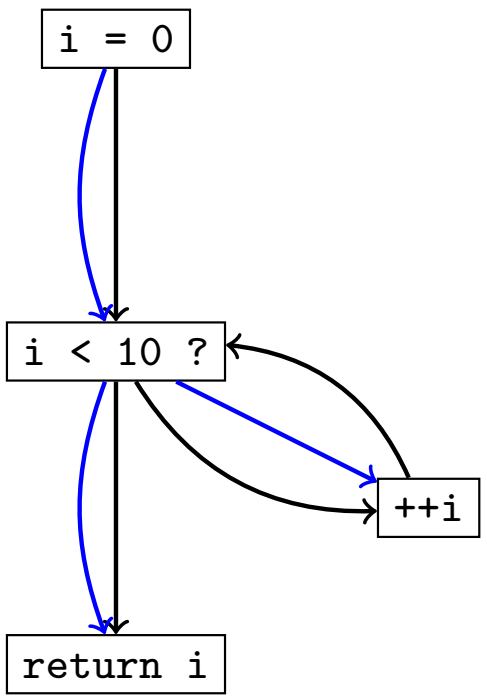
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

postdominacja

to dominacja od t po odwróconych krawędziach



Dominacja

u dominuje v

gdy każda ścieżka z s do v przechodzi przez u

u bezpośrednio dominuje v

gdy dodatkowo u nie dominuje żadnego innego dominatora v

bezpośrednie dominatory

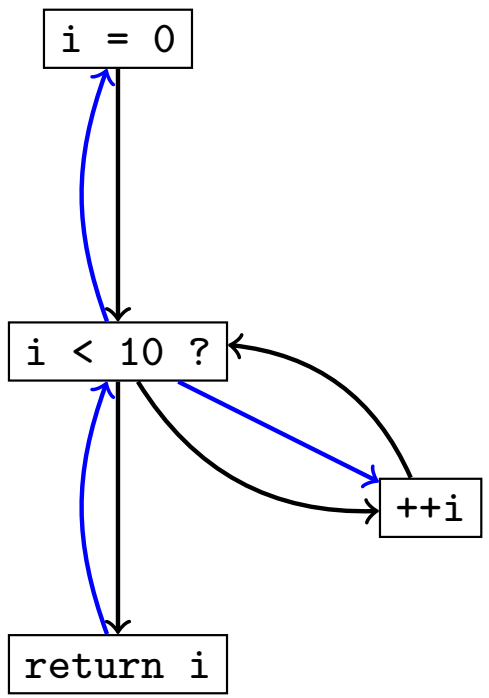
- tworzą drzewo o korzeniu w s
- można wyznaczyć w czasie prawie liniowym

$v \in$ dominance frontier(u)

- $u \rightsquigarrow w \longrightarrow v$
- u dominuje w
- u nie dominuje v

postdominacja

to dominacja od t po odwróconych krawędziach



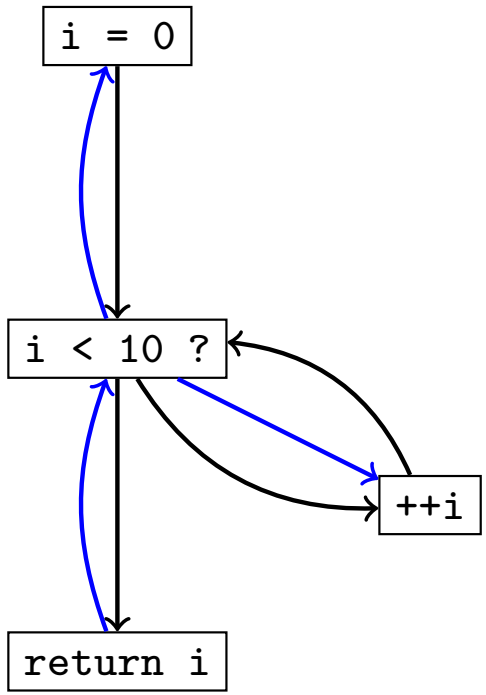
Control Dependence Graph

v control-depends on u

- $\exists u \longrightarrow w \rightsquigarrow v$
- v post-dominuje w (lub $v = w$)
- v nie post-dominuje u

Intuicja

- u ma przynajmniej 2 wyjścia
- jedno z nich zawsze prowadzi do v
- drugie nie



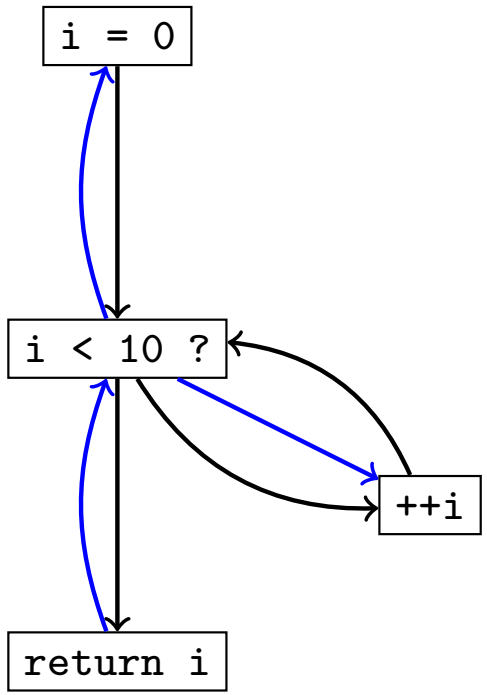
Control Dependence Graph

v control-depends on u

- $\exists u \longrightarrow w \rightsquigarrow v$
- v post-dominuje w (lub $v = w$)
- v nie post-dominuje u

Intuicja

- u ma przynajmniej 2 wyjścia
- jedno z nich zawsze prowadzi do v
- drugie nie



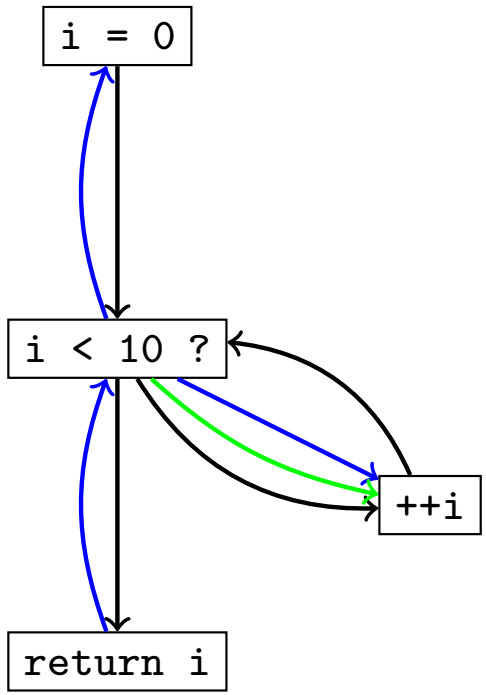
Control Dependence Graph

v control-depends on u

- $\exists u \longrightarrow w \rightsquigarrow v$
- v post-dominuje w (lub $v = w$)
- v nie post-dominuje u

Intuicja

- u ma przynajmniej 2 wyjścia
- jedno z nich zawsze prowadzi do v
- drugie nie



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

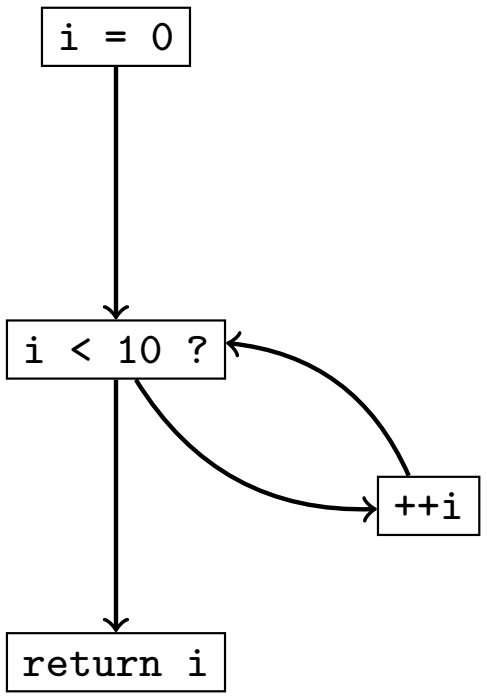
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

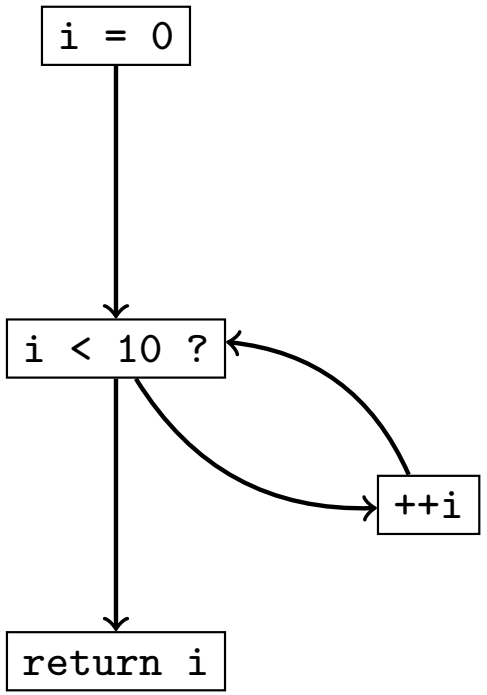
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

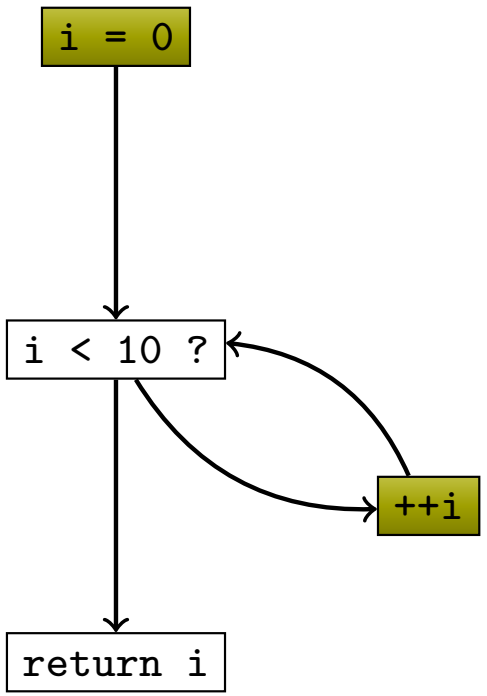
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

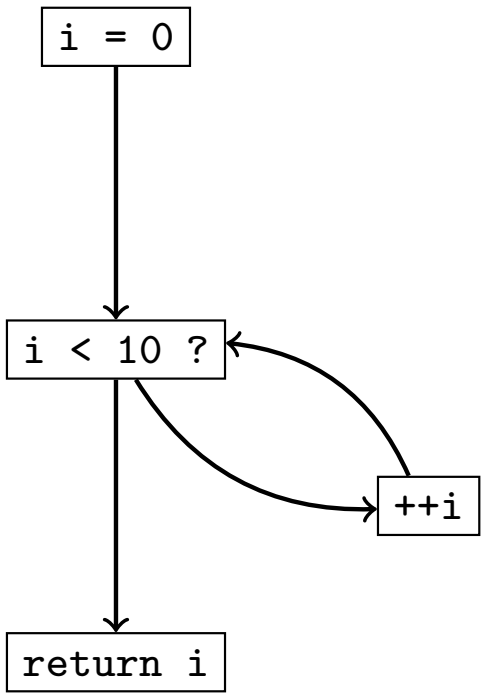
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

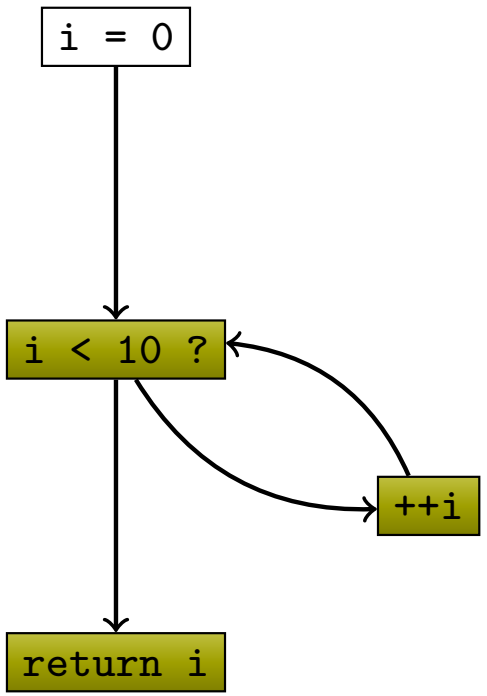
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

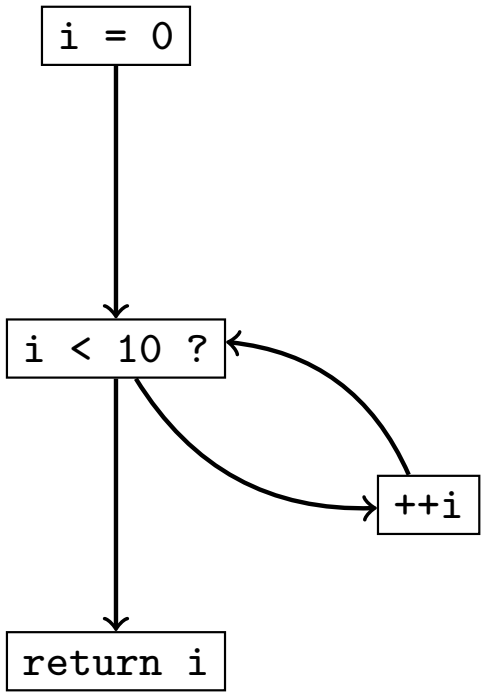
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

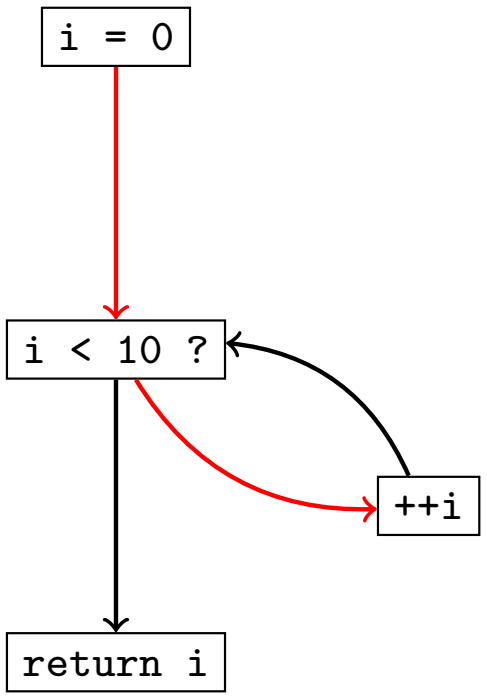
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

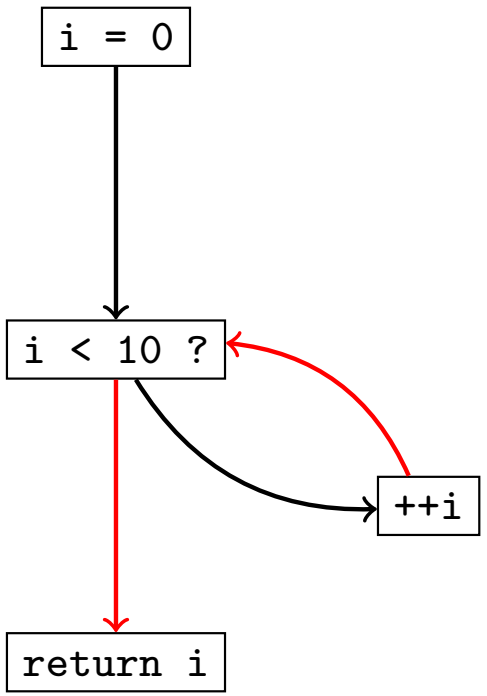
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

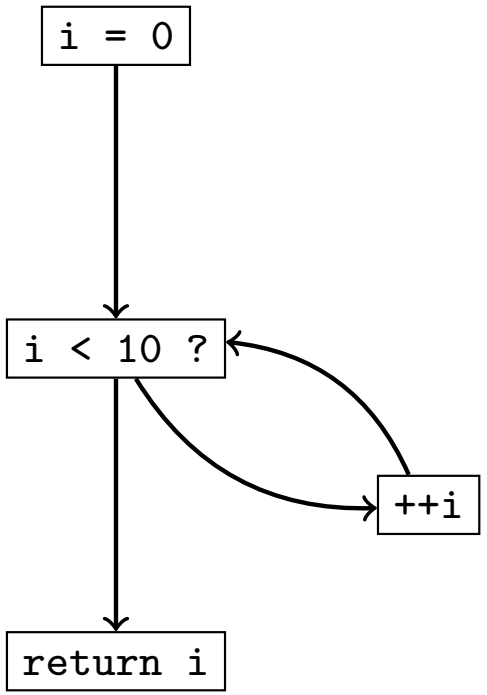
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

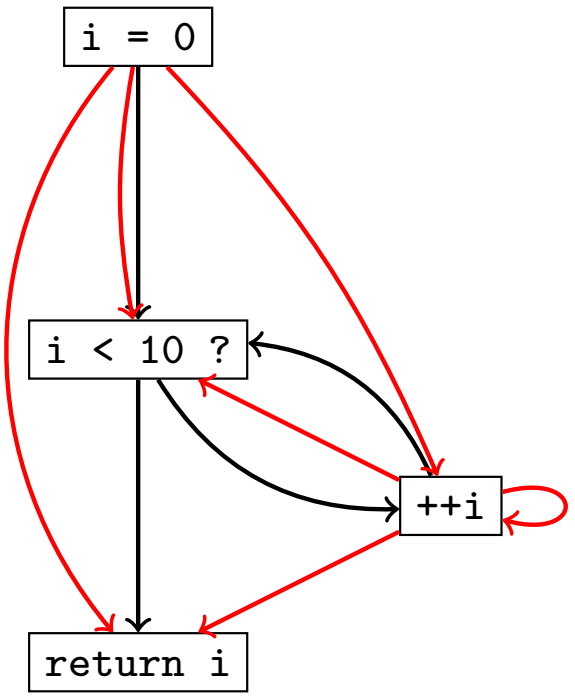
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x



Data Flow (Dependence) Graph

Definicja zmiennej x

to instrukcja ustawiająca x

Użycie zmiennej x

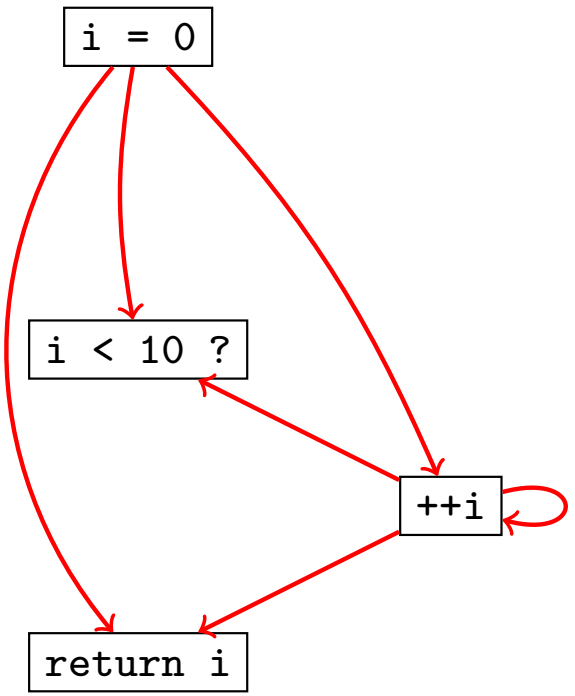
to instrukcja odczytująca x

Ścieżka wolna dla x

to ścieżka w CFG omijająca definicje x

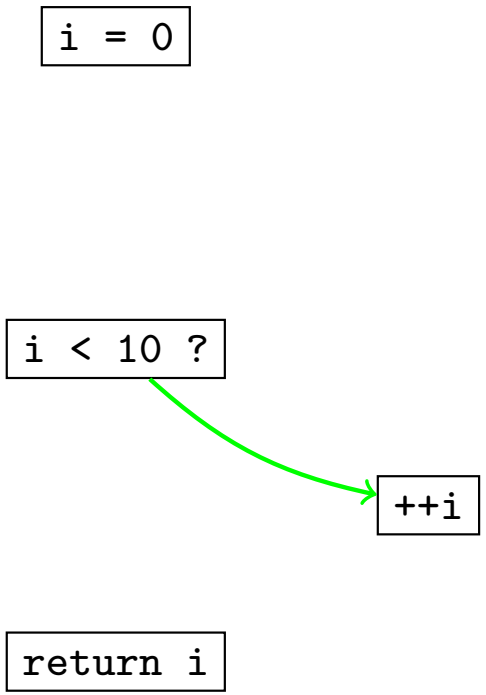
Krawędź $u \to v$ w DFG

- u – definicja x
- v – użycie x
- $\exists u \rightsquigarrow v$ wolna dla x

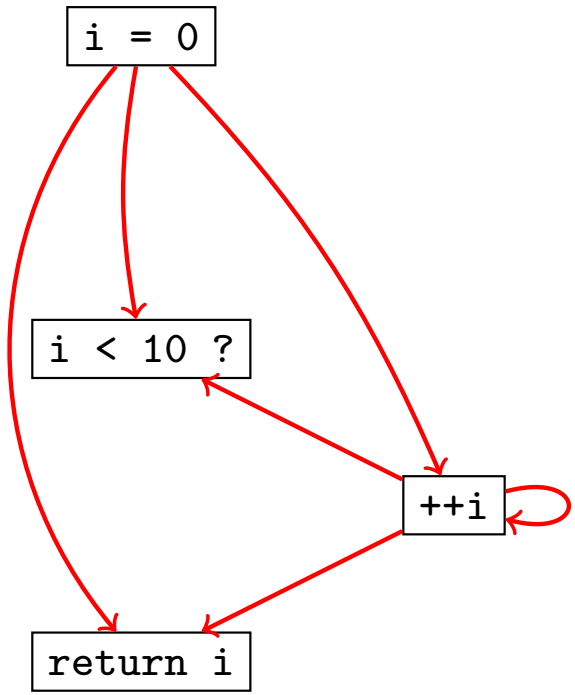


Program Dependence Graph

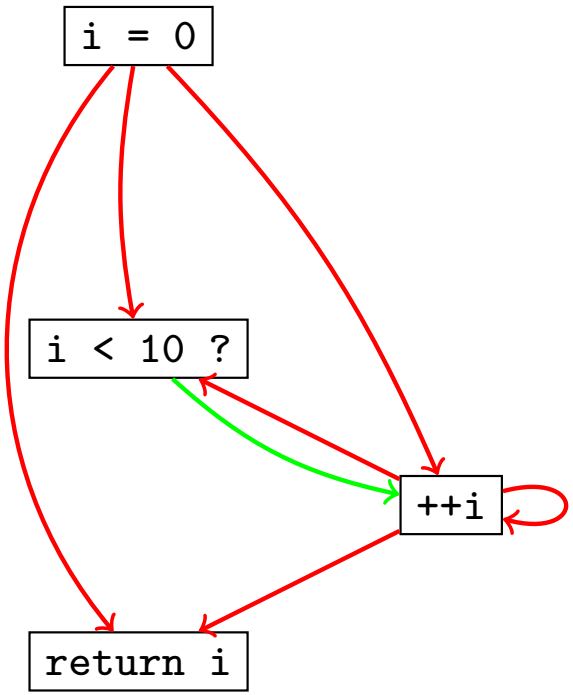
Control Dependence



Data Dependence



Program Dependence Graph



Static Single Assignment form

Definicja zmiennej x

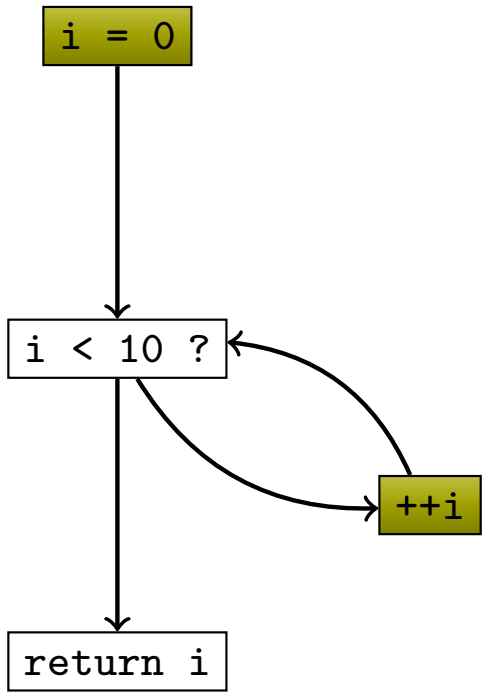
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



Static Single Assignment form

Definicja zmiennej x

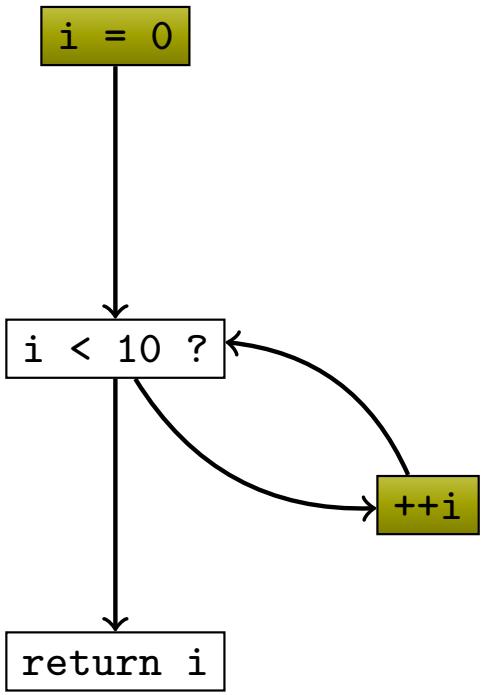
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



Static Single Assignment form

Definicja zmiennej x

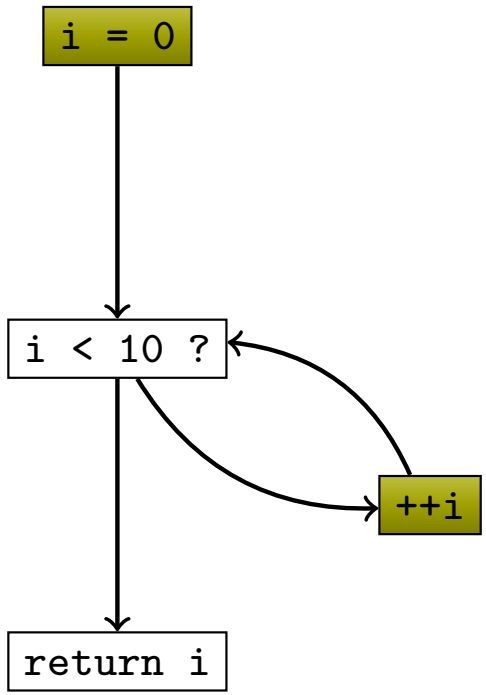
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



Static Single Assignment form

Definicja zmiennej x

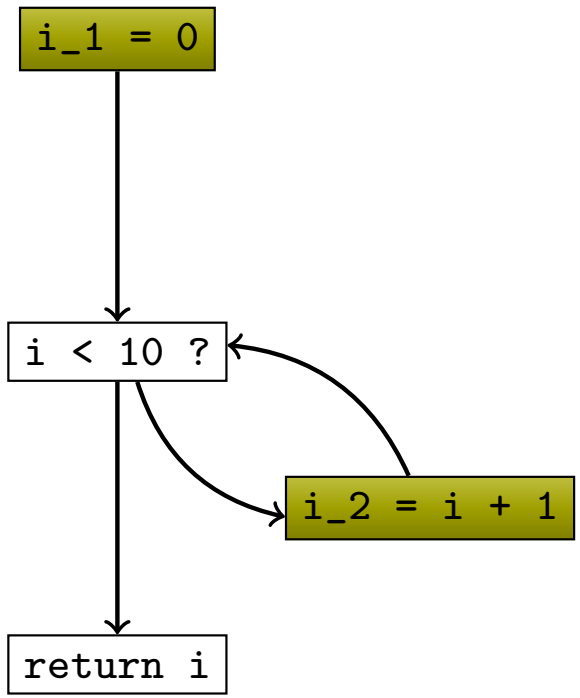
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



Static Single Assignment form

Definicja zmiennej x

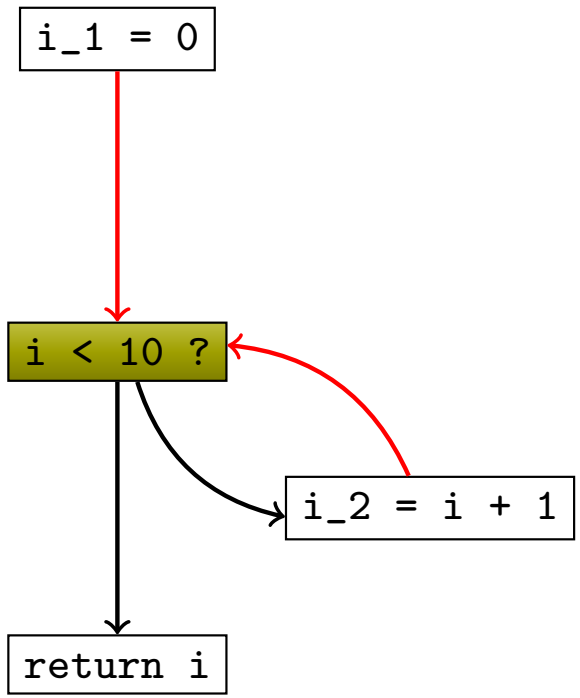
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



Static Single Assignment form

Definicja zmiennej x

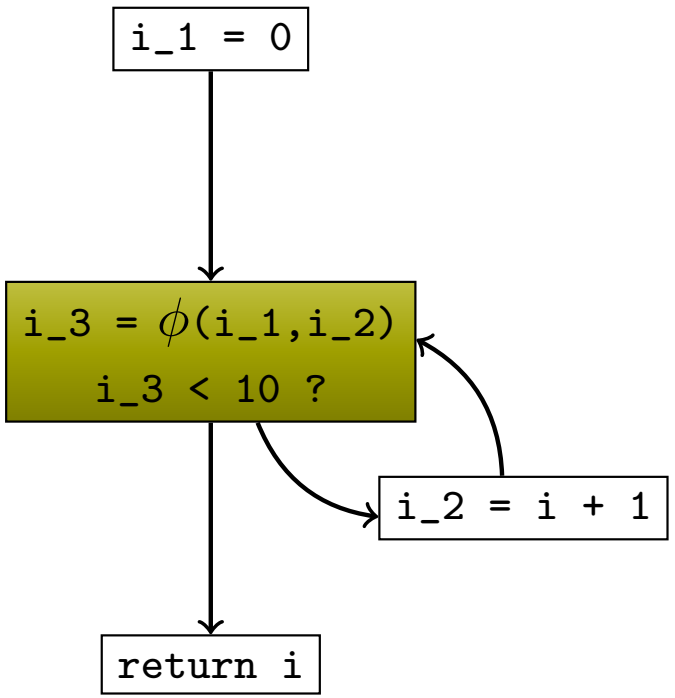
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



Static Single Assignment form

Definicja zmiennej x

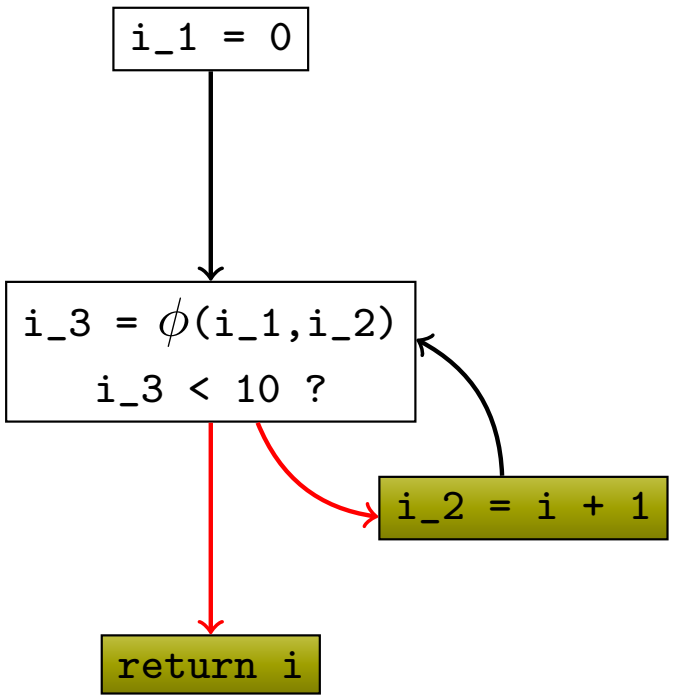
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



Static Single Assignment form

Definicja zmiennej x

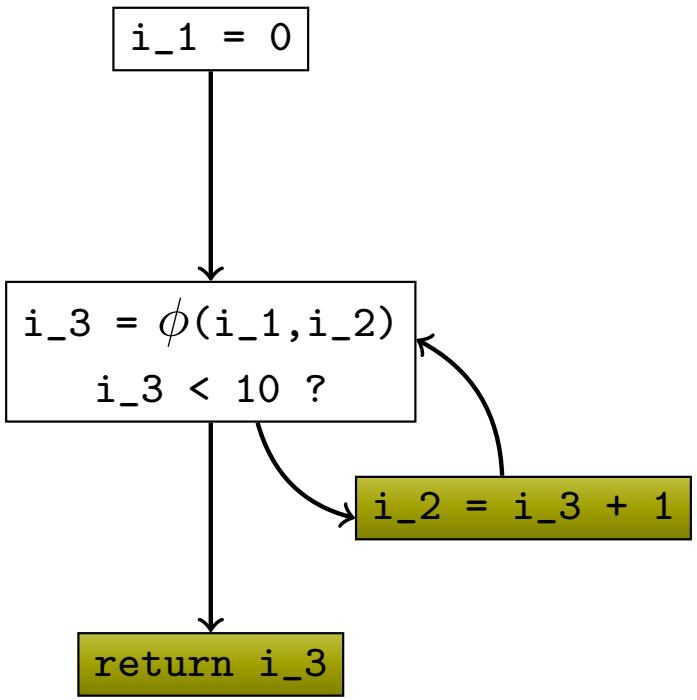
to instrukcja ustawiająca x

Ograniczenie SSA

każda zmienna ma dokładnie 1 definicję

Konstrukcja

- nowa nazwa zmiennej w każdej definicji
- spotkanie 2+ definicji – sztuczna zmienna
- propagacja nowych nazw



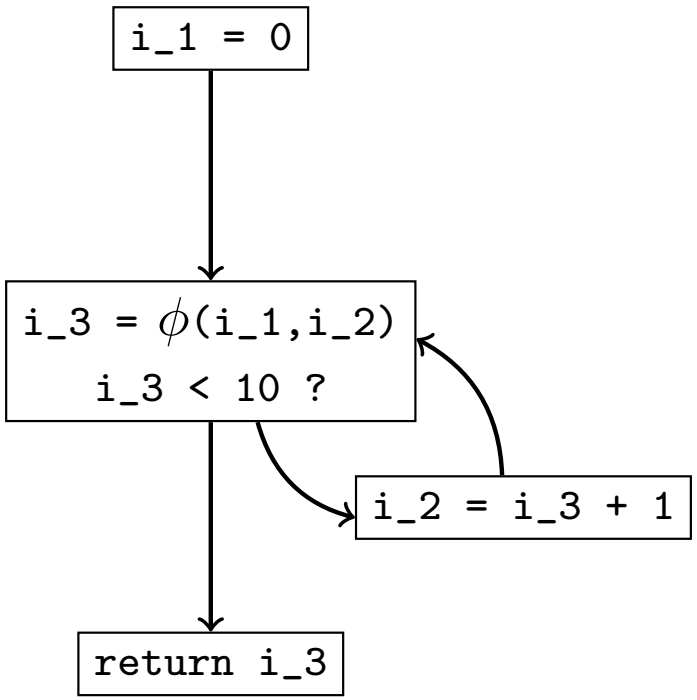
Static Single Information form

Ograniczenia SSA

- każda zmienna ma dokładnie 1 definicję
- definicja zmiennej dominuje każde użycie
- każde użycie zmiennej post-dominuje definicję
- ϕ-użycia są w dominance frontier definicji
- σ-definicje są w postdominance frontier użyć

Konstrukcja

- dla każdej niezależnej gałęzi – sztuczna zmienna
- propagacja nowych nazw



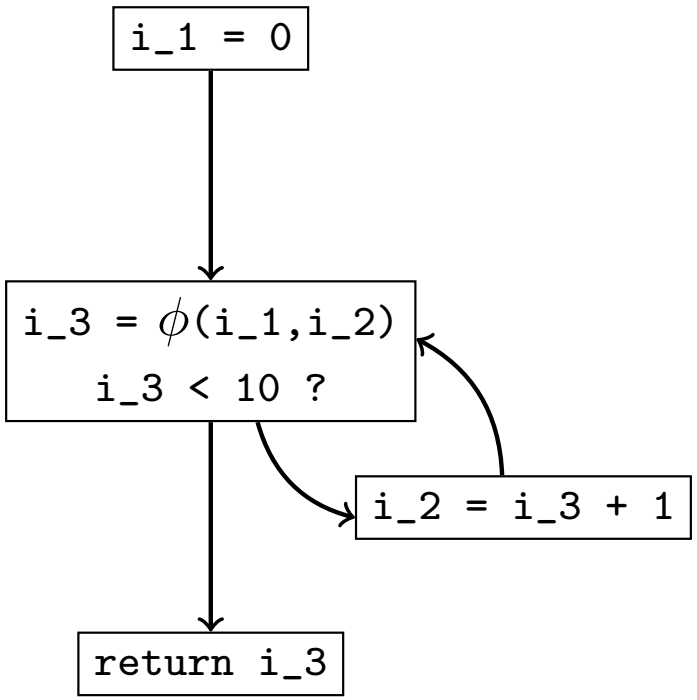
Static Single Information form

Ograniczenia SSI

- każda zmienna ma dokładnie 1 definicję
- definicja zmiennej dominuje każde użycie
- każde użycie zmiennej post-dominuje definicję
- ϕ-użycia są w dominance frontier definicji
- σ-definicje są w postdominance frontier użyć

Konstrukcja

- dla każdej niezależnej gałęzi – sztuczna zmienna
- propagacja nowych nazw



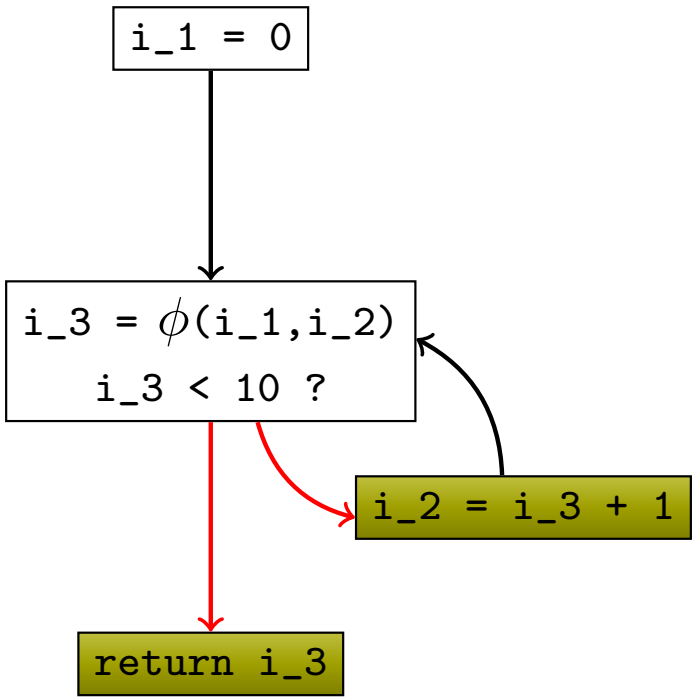
Static Single Information form

Ograniczenia SSI

- każda zmienna ma dokładnie 1 definicję
- definicja zmiennej dominuje każde użycie
- każde użycie zmiennej post-dominuje definicję
- ϕ-użycia są w dominance frontier definicji
- σ-definicje są w postdominance frontier użyć

Konstrukcja

- dla każdej niezależnej gałęzi – sztuczna zmienna
- propagacja nowych nazw



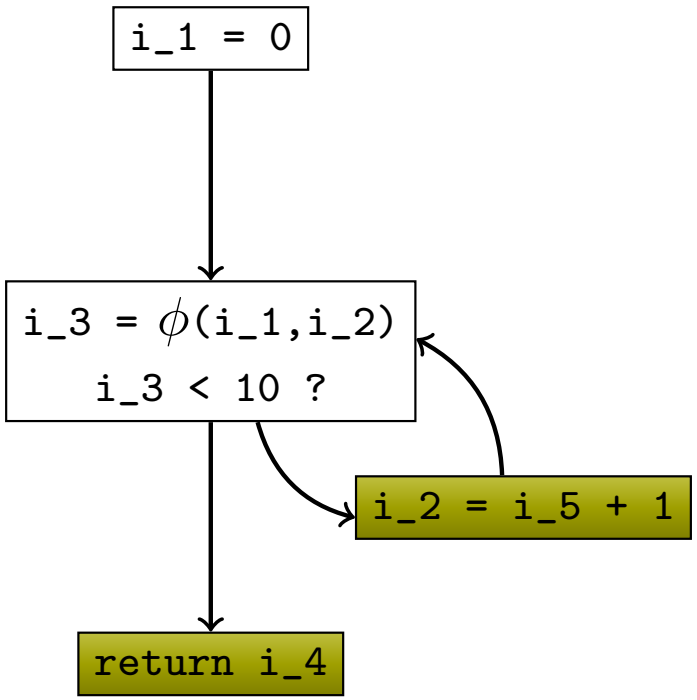
Static Single Information form

Ograniczenia SSI

- każda zmienna ma dokładnie 1 definicję
- definicja zmiennej dominuje każde użycie
- każde użycie zmiennej post-dominuje definicję
- ϕ-użycia są w dominance frontier definicji
- σ-definicje są w postdominance frontier użyć

Konstrukcja

- dla każdej niezależnej gałęzi – sztuczna zmienna
- propagacja nowych nazw



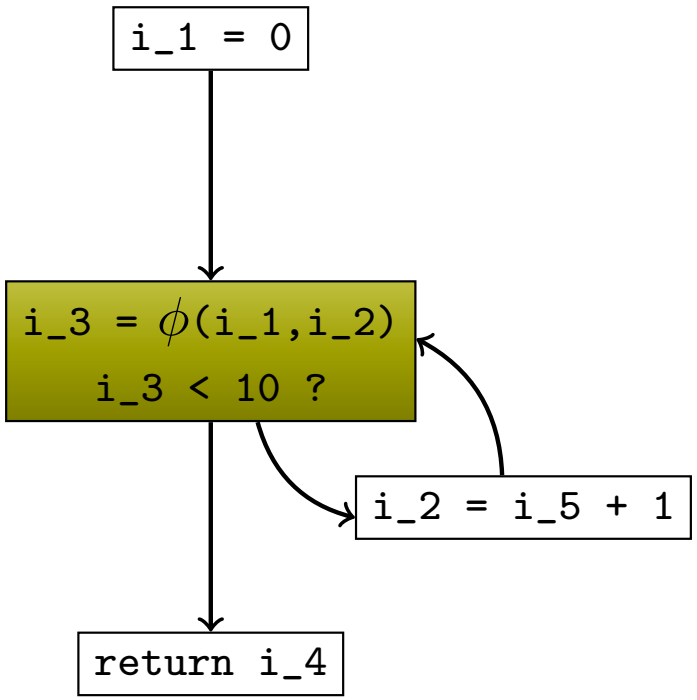
Static Single Information form

Ograniczenia SSI

- każda zmienna ma dokładnie 1 definicję
- definicja zmiennej dominuje każde użycie
- każde użycie zmiennej post-dominuje definicję
- ϕ-użycia są w dominance frontier definicji
- σ-definicje są w postdominance frontier użyć

Konstrukcja

- dla każdej niezależnej gałęzi – sztuczna zmienna
- propagacja nowych nazw



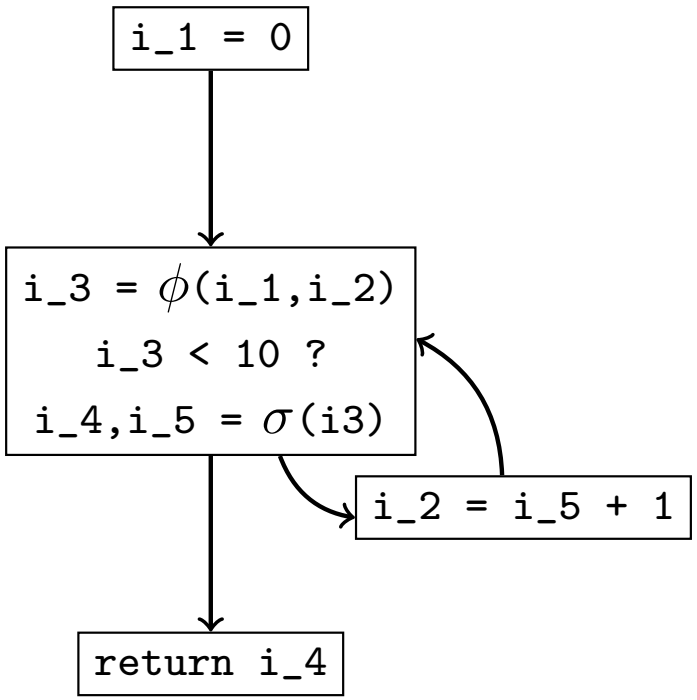
Static Single Information form

Ograniczenia SSI

- każda zmienna ma dokładnie 1 definicję
- definicja zmiennej dominuje każde użycie
- każde użycie zmiennej post-dominuje definicję
- ϕ-użycia są w dominance frontier definicji
- σ-definicje są w postdominance frontier użyć

Konstrukcja

- dla każdej niezależnej gałęzi – sztuczna zmienna
- propagacja nowych nazw



Static Single Information form

Ograniczenia SSI

- każda zmienna ma dokładnie 1 definicję
- definicja zmiennej dominuje każde nie-ϕ-użycie
- każde użycie zmiennej post-dominuje nie-σ-definicję
- ϕ-użycia są w dominance frontier definicji
- σ-definicje są w postdominance frontier użyć

Konstrukcja

- dla każdej niezależnej gałęzi – sztuczna zmienna
- propagacja nowych nazw