# Weryfikacja wspomagana komputerowo

wykład 11

## Interpretacja abstrakcyjna I

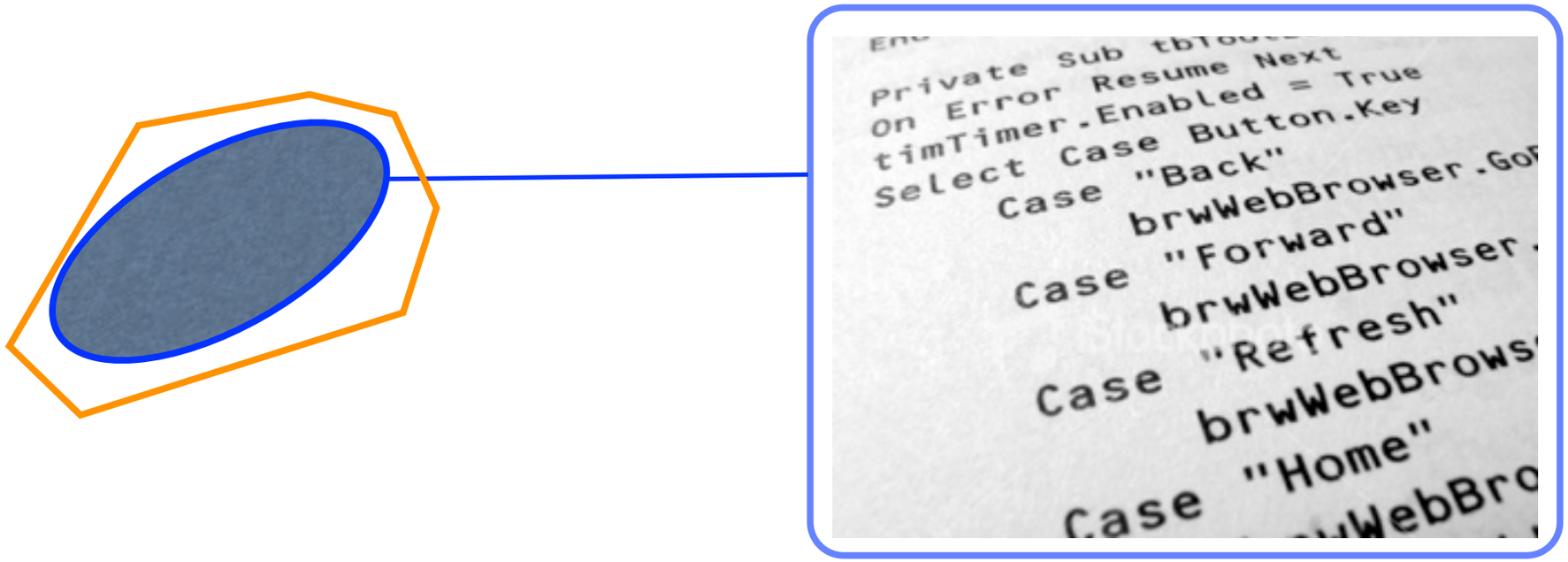
# Źródła

- F. Nielson, H.R. Nielson, C. Hankin, Principles of Program Analysis, Springer, 2005.
- http://www.imm.dtu.dk/~riis/PPA/slides2.pdf
- V. D'Silva, D. Kroening, G. Weissenbacher, A Survey of Automated Techniques for Formal Software Verification. IEEE Trans. on CAD of Integrated Circuits and Systems 27 (7):1165-1178, 2008.

# Pionierzy

- P. Naur 1965
- P. Cousot, R. Cousot 1977

# Analiza przybliżona




na pewno ✔

chyba ✘

# Interpretacja abstrakcyjna

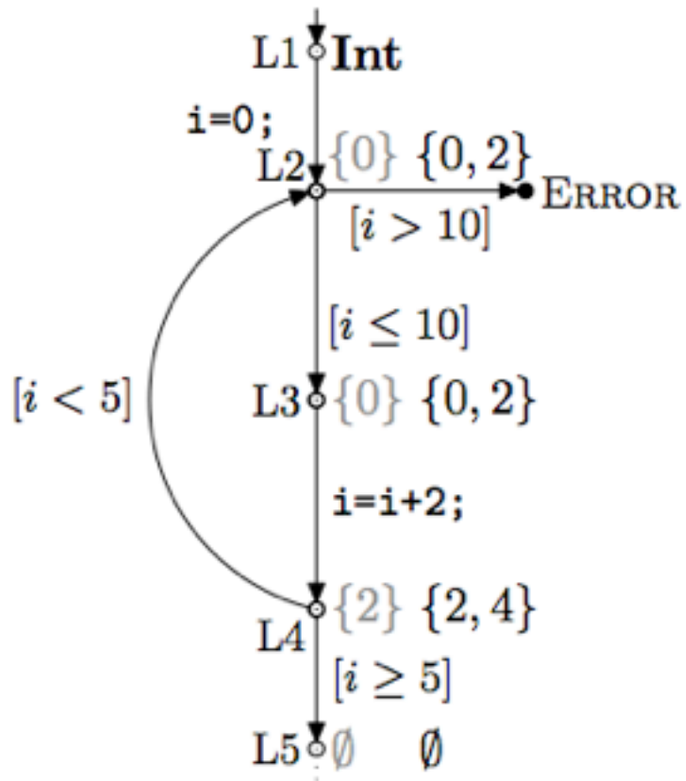
$$123 \cdot 457 + 76543 \stackrel{?}{=} 132654$$

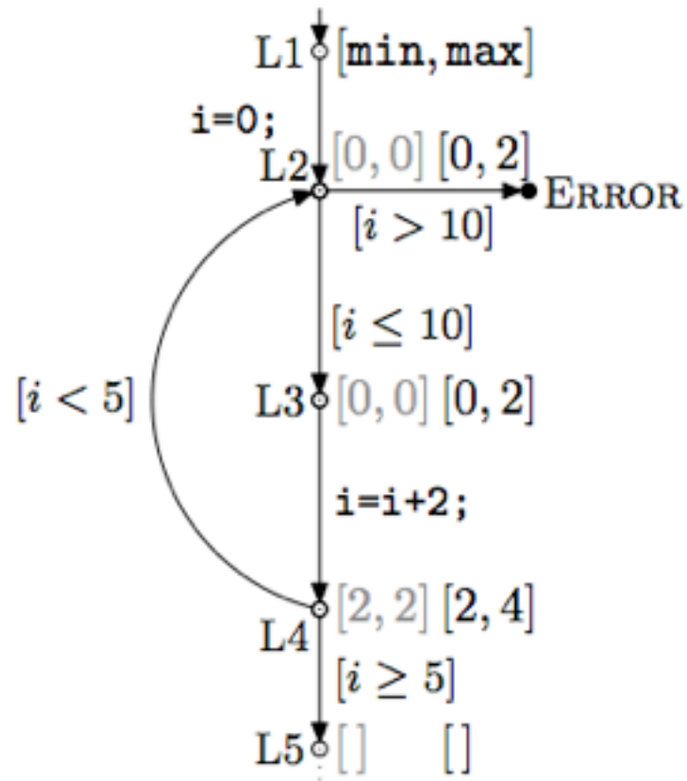
$$123 \cdot 457 + 76543 \stackrel{?}{=} 132654$$

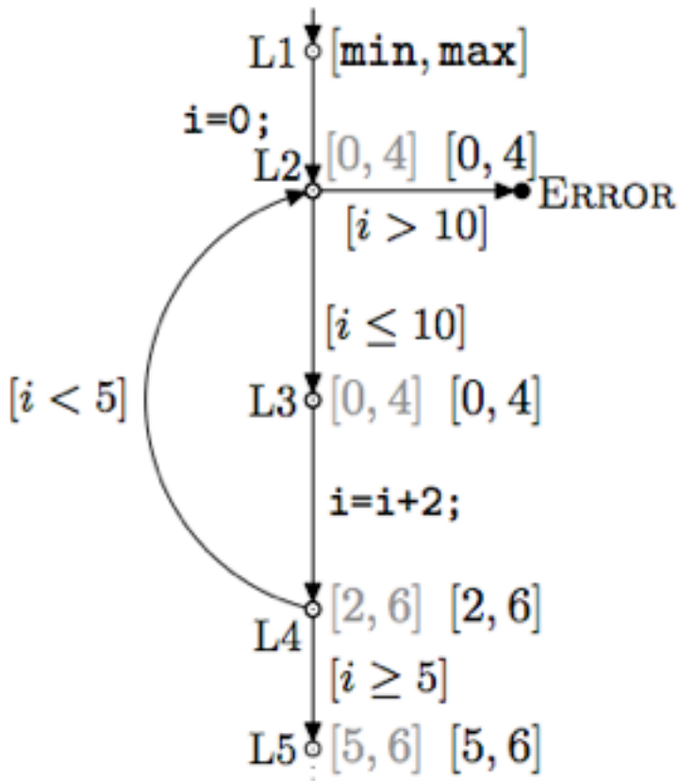
$$6 \cdot 7 + 7 \stackrel{?}{=} 3 \pmod 9$$

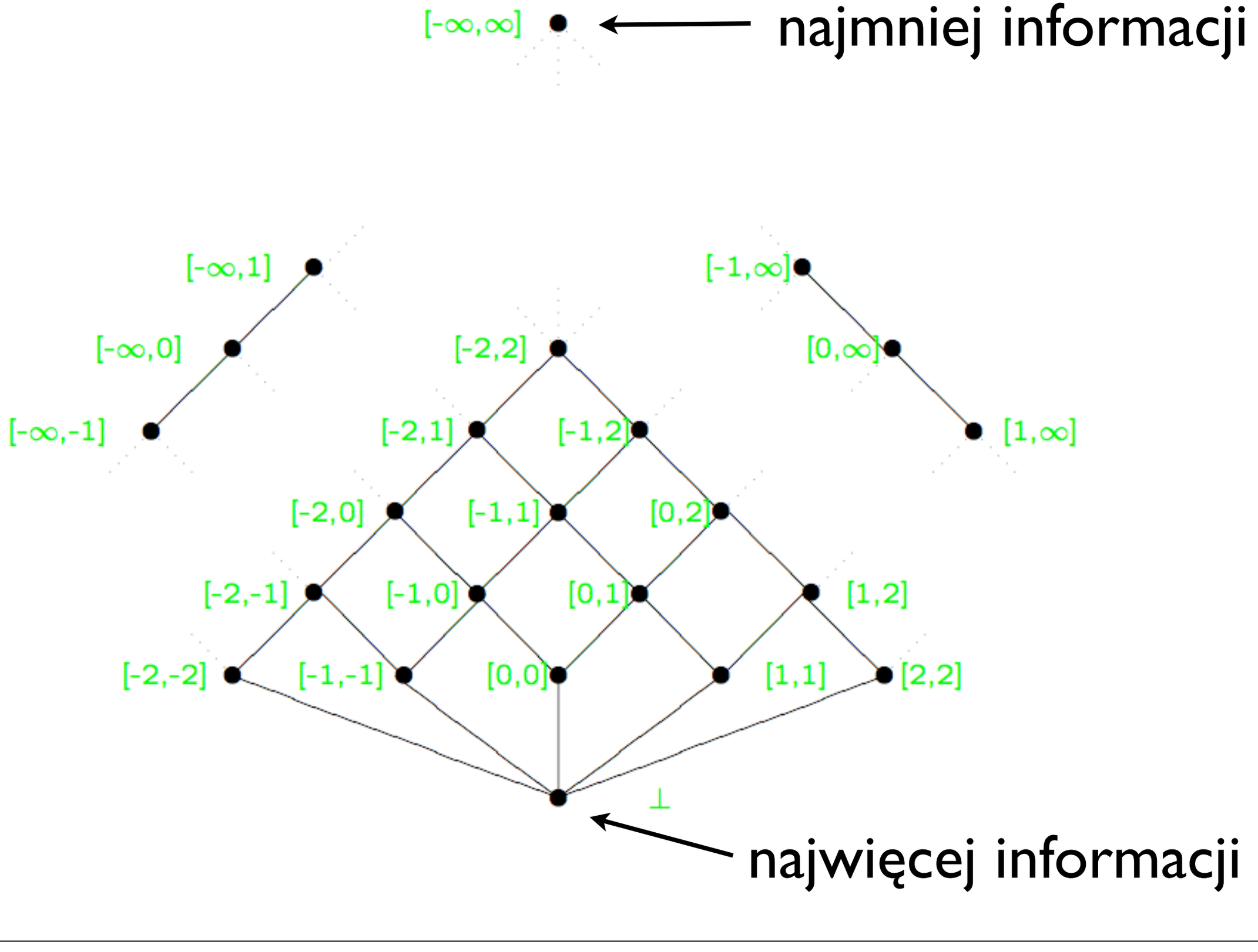
$$6 + 7 \stackrel{?}{=} 3 \pmod 9$$

```
int i = 0;
do {
   assert(i <= 10);
   i = i+2;
} while (i < 5);
```




[-∞,∞]
najmniej informacji
[-∞,1]
[-∞,0]
[-∞,-1]
[-2,2]
[-2,1]
[-1,2]
[-2,0]
[-1,1]
[0,2]
[-2,-1]
[-1,0]
[0,1]
[1,2]
[-2,-2]
[-1,-1]
[0,0]
[1,1]
[2,2]
[-1,∞]
[0,∞]
[1,∞]
⊥
najwięcej informacji

# Analiza przybliżona

- analizujemy program źródłowy (diagram przepływu sterowania)
- fałszywe alarmy (ang. false positives)
- na ogół ukierunkowana na specyficzną własność
- w pełni automatyczna
- stosowalna do programów dużego rozmiaru
- gdy odp. negatywna, informacja diagnostyczna

# Analiza przybliżona - metody

- analiza przepływu danych
- analiza przepływu sterowania
- analiza typów
- analiza efektywności
- ...
- interpretacja abstrakcyjna

# Analiza przybliżona - zastosowania

- kompilacja, optymalizacja kodu
- weryfikacja poprawności programów
- weryfikacja jakości kodu
- interpretacja abstrakcyjna - systematyzacja:
  - ...
  - statyczne typowanie
  - abstrakcja w model checking'u

# Optymalizacja kodu

- propagacja stałych (obliczanie w czasie kompilacji)
- propagacja kopii
- dostępne wyrażenia (eliminacja obliczeń)
- żywe zmienne (eliminacja martwego kodu)
- definicja - użycie, użycie - definicja
- analiza ścisłości
- zakres tablic
- ...

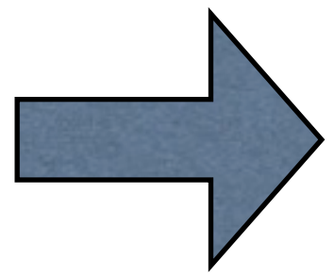
# Weryfikacja programów

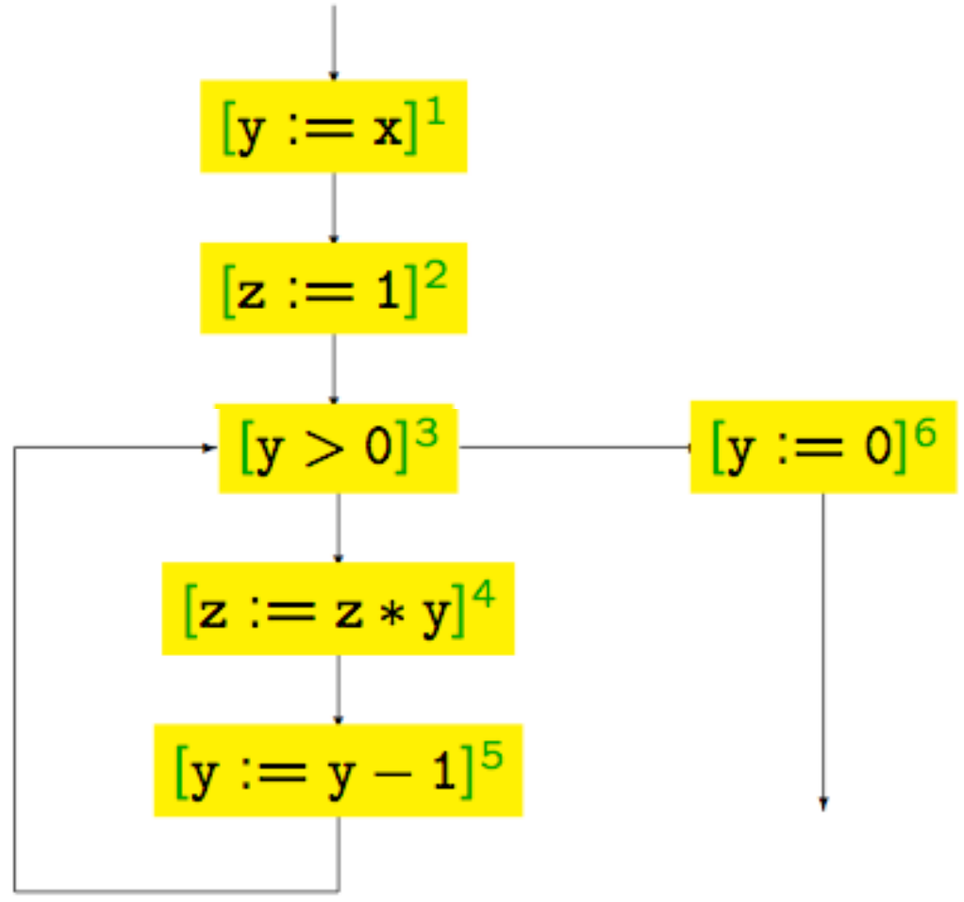
- dzielenie przez 0
- wskaźniki
  - NULL
  - dane statyczne i dynamiczne (stos i sterta)
    - analiza kształtu
- aliasy (procedury)
- zakres tablic
- wykrywanie niezmienników
- ...

# Analiza przepływu danych

# Analiza przepływu danych

```
[y := x]1;
[z := 1]2;
while [y > 0]3 do
    [z := z * y]4;
    [y := y − 1]5
od;
[y := 0]6
```






(while-programy)

skończony zbiór punktów sterowania

$$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$$

$$\mathrm{init}(S) \subseteq S$$

$$\mathrm{State} = S \times \mathrm{Store}$$

$$\mathrm{Store} = \mathrm{Var} \to \mathrm{Val}$$

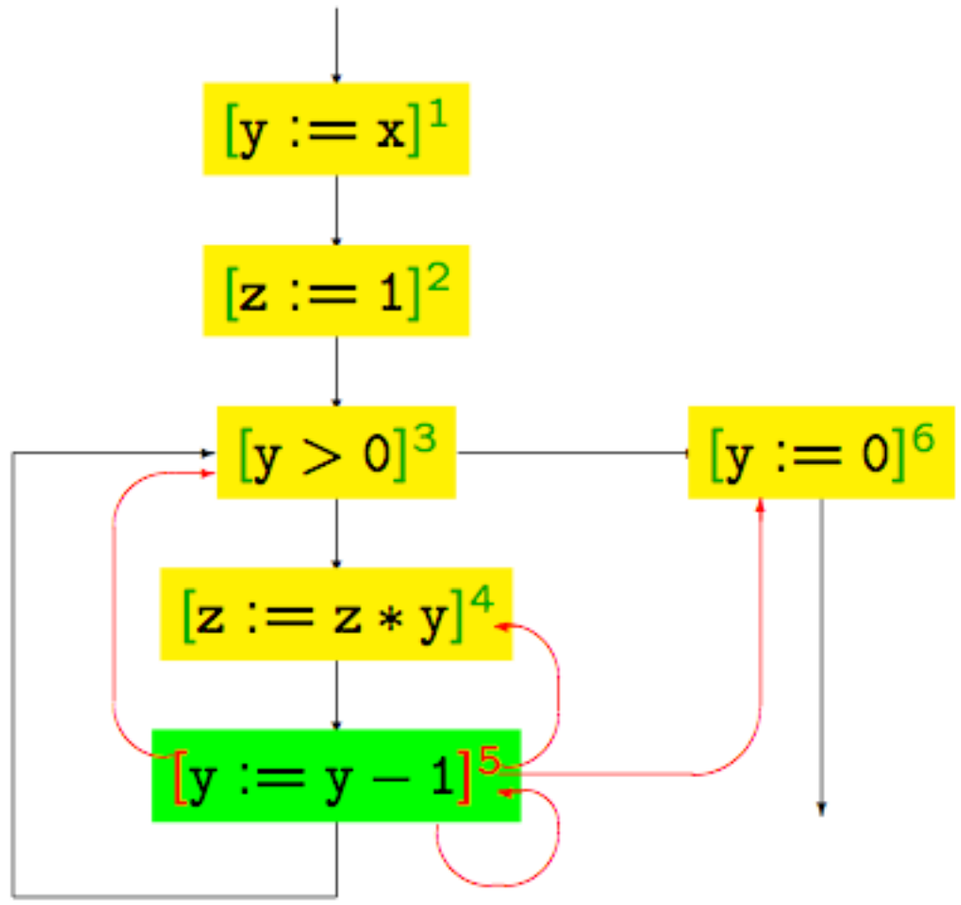
# Analiza przepływu danych

- „docierające” przypisania
- dostępne wyrażenia
- żywe zmienne
- niezbędne wyrażenia
- propagacja stałych
- ...

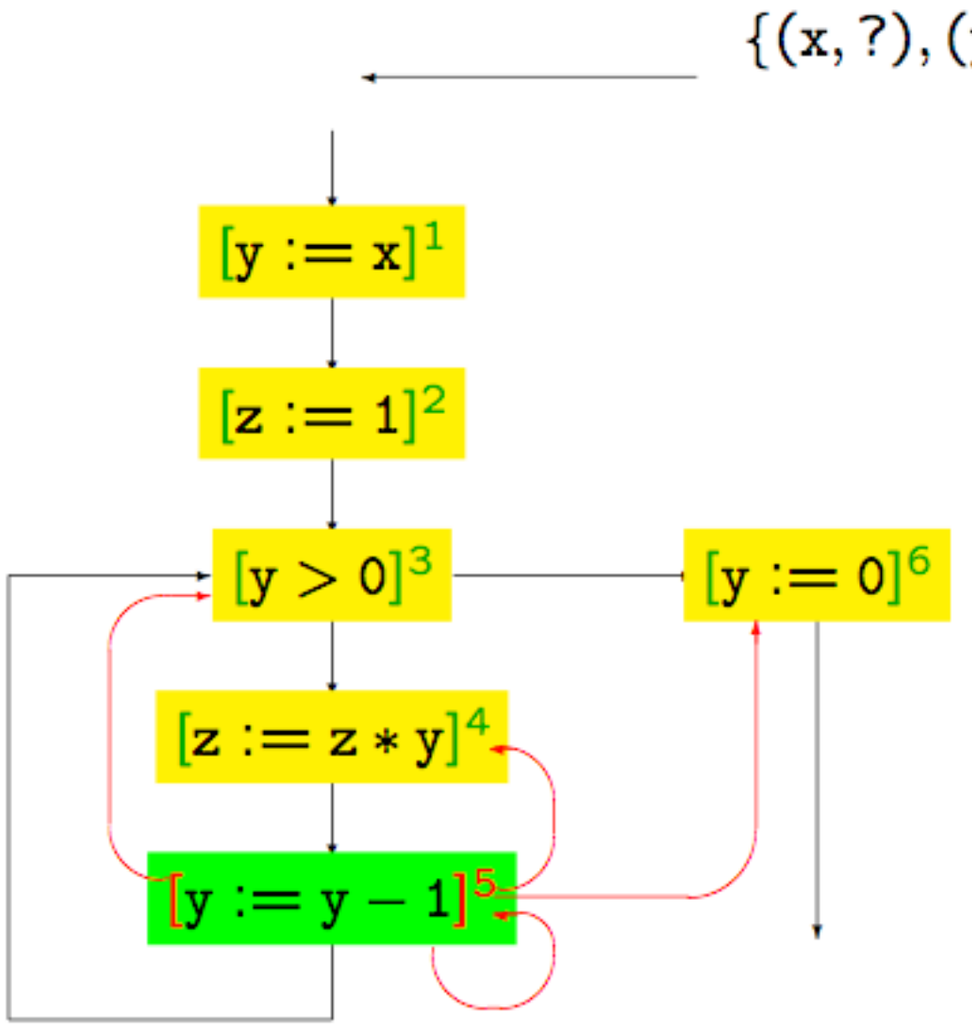
# „Docierające” przypisania

W każdym punkcie sterowania oblicz zbiór przypisań, które zostały wykonane (i nie „nadpisane”) zanim program osiągnął ten punkt.

# „Docierające” przypisania



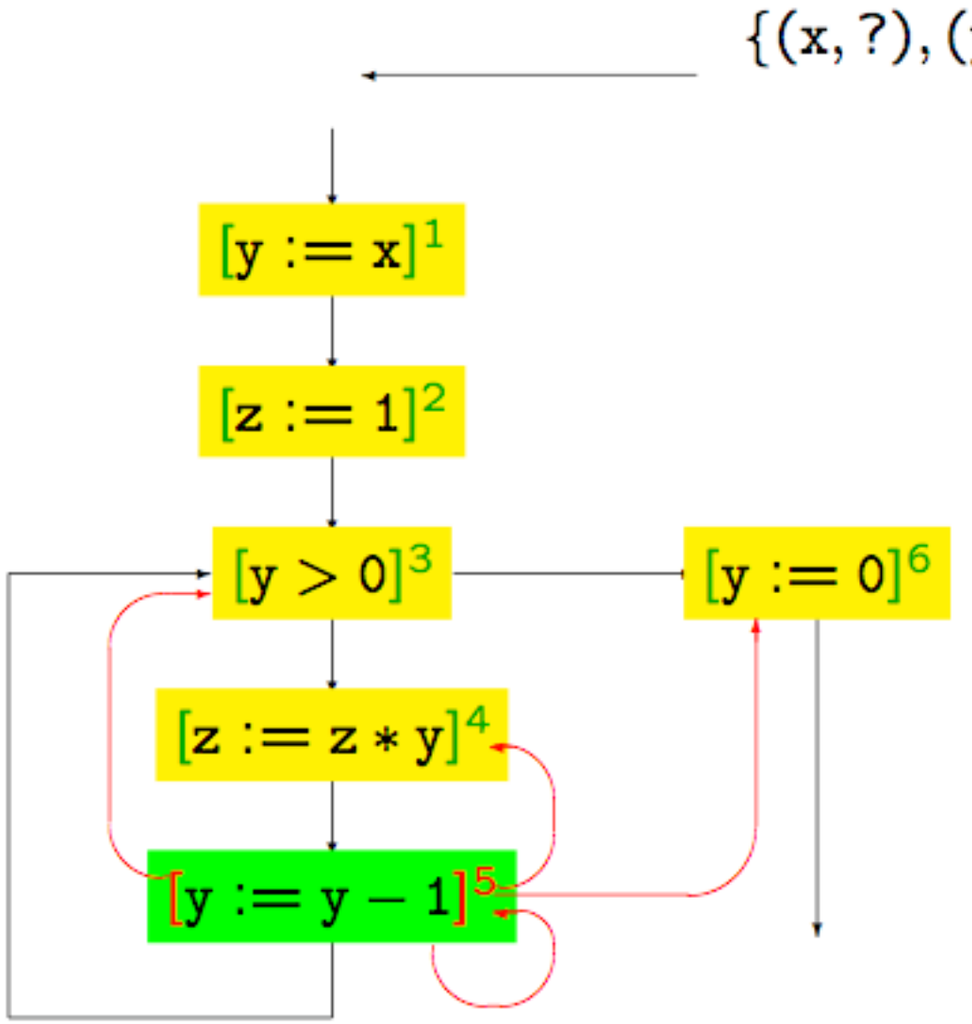
# „Docierające” przypisania




kill(i) gen(i) ∪

- formalizujemy problem jako układ równań
  - zmienne reprezentują informację przed/po instrukcji
- najmniejsze rozwiązanie
- algorytm iteracyjny

# „Docierające” przypisania




$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcup_{j \rightsquigarrow i} x_j^{\text{exit}}$$

$$x_1^{\text{entry}} = \{(\mathtt{x}, ?), (\mathtt{y}, ?), (\mathtt{z}, ?)\}$$

interesuje nas najmniejsze rozwiązanie

# „Docierające” przypisania

$\mathrm{kill}(\mathrm{z} := \mathrm{e}) = \{(\mathrm{z}, ?)\} \cup \{(\mathrm{z}, j) \mid j \in S\}$

$\mathrm{kill}(\mathrm{b}) = \emptyset$

$\mathrm{kill}(\mathrm{skip}) = \emptyset$

$\mathrm{gen}([\mathrm{z} := \mathrm{e}]^i) = \{(\mathrm{z}, i)\}$

$\mathrm{gen}(\mathrm{b}) = \emptyset$

$\mathrm{gen}(\mathrm{skip}) = \emptyset$

$$x_i^{\mathrm{exit}} = x_i^{\mathrm{entry}} \setminus \mathrm{kill}(i) \cup \mathrm{gen}(i)$$

$$x_i^{\mathrm{entry}} = \bigcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}}$$

$$x_1^{\mathrm{entry}} = \{(\mathtt{x}, ?), (\mathtt{y}, ?), (\mathtt{z}, ?)\}$$

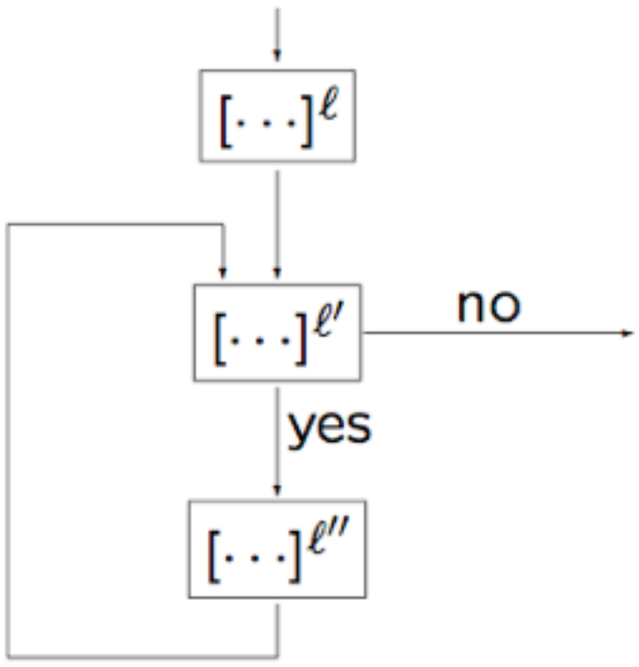
interesuje nas najmniejsze rozwiązanie

# „Docierające” przypisania

```
                                    ←—— {(x,?),(y,?),(z,?)}
[y := x]^1;                         ←—— {(x,?),(y,1),(z,?)}
[z := 1]^2;                         ←—— {(x,?),(y,1),(y,5),(z,2),(z,4)}
while [y > 0]^3 do                  ←—— {(x,?),(y,1),(y,5),(z,2),(z,4)}
    [z := z * y]^4;                 ←—— {(x,?),(y,1),(y,5),(z,2),(z,4)}
    [y := y - 1]^5                  ←—— {(x,?),(y,1),(y,5),(z,2),(z,4)}
od;                                 ←—— {(x,?),(y,1),(y,5),(z,2),(z,4)}
[y := 0]^6                          ←—— {(x,?),(y,6),(z,2),(z,4)}
```

# „Docierające” przypisania

$[\texttt{z:=x+y}]^{\ell}; \texttt{while } [\texttt{true}]^{\ell'} \texttt{ do } [\texttt{skip}]^{\ell''}$



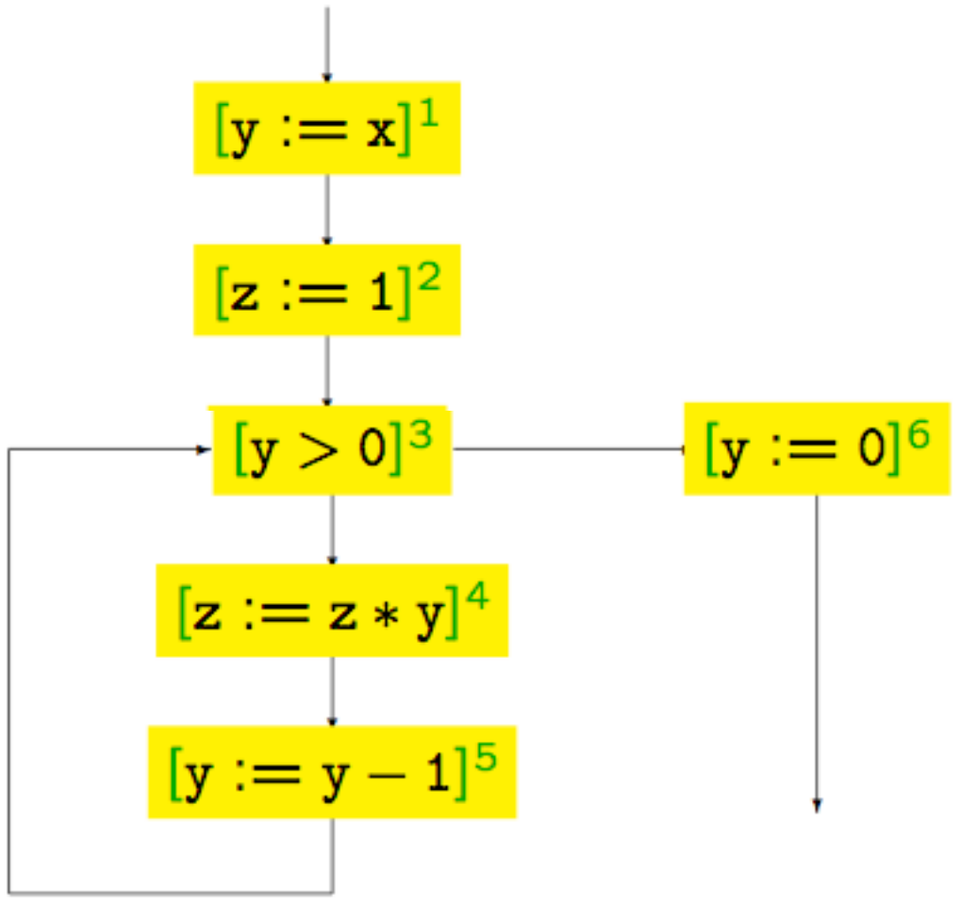

**rozwiązania:** $x_{l'}^{\text{entry}} \supseteq \{(\texttt{x}, ?), (\texttt{y}, ?), (\texttt{z}, \texttt{1})\}$

interesuje nas najmniejsze rozwiązanie

# Dostępne wyrażenia

W każdym punkcie sterowania oblicz zbiór wyrażeń, których wartości są na pewno obliczone przed wejściem do tego punktu.

# Dostępne wyrażenia




$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcap_{j \rightsquigarrow i} x_j^{\text{exit}}$$

$$x_1^{\text{entry}} = \emptyset$$

interesuje nas największe rozwiązanie

# Dostępne wyrażenia

[x:=a+b]$^1$; [y:=a*b]$^2$; while [y> a+b ]$^3$ do ([a:=a+1]$^4$; [x:=a+b]$^5$)

| $i$ | $x_i^{\text{entry}}$ | $x_i^{\text{exit}}$ |
|---|---|---|
| 1 | $\emptyset$ | {a+b} |
| 2 | {a+b} | {a+b, a*b} |
| 3 | {a+b} | {a+b} |
| 4 | {a+b} | $\emptyset$ |
| 5 | $\emptyset$ | {a+b} |

$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

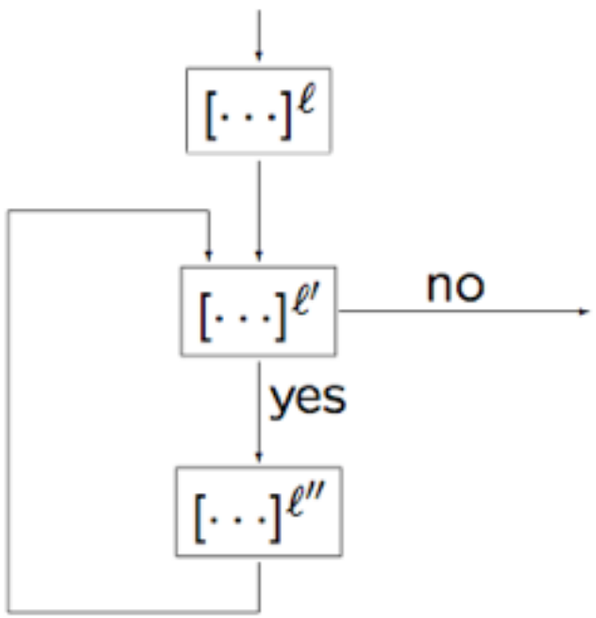
$$x_i^{\text{entry}} = \bigcap_{j \rightsquigarrow i} x_j^{\text{exit}}$$

$$x_1^{\text{entry}} = \emptyset$$

interesuje nas największe rozwiązanie

# Dostępne wyrażenia

$[\texttt{z:=x+y}]^{\ell}; \texttt{while } [\texttt{true}]^{\ell'} \texttt{ do } [\texttt{skip}]^{\ell''}$




$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcap_{j \rightsquigarrow i} x_j^{\text{exit}}$$

dwa rozwiązania:

$$x_{l'}^{\text{entry}} = \{x+y\}, \emptyset$$

$$x_1^{\text{entry}} = \emptyset$$

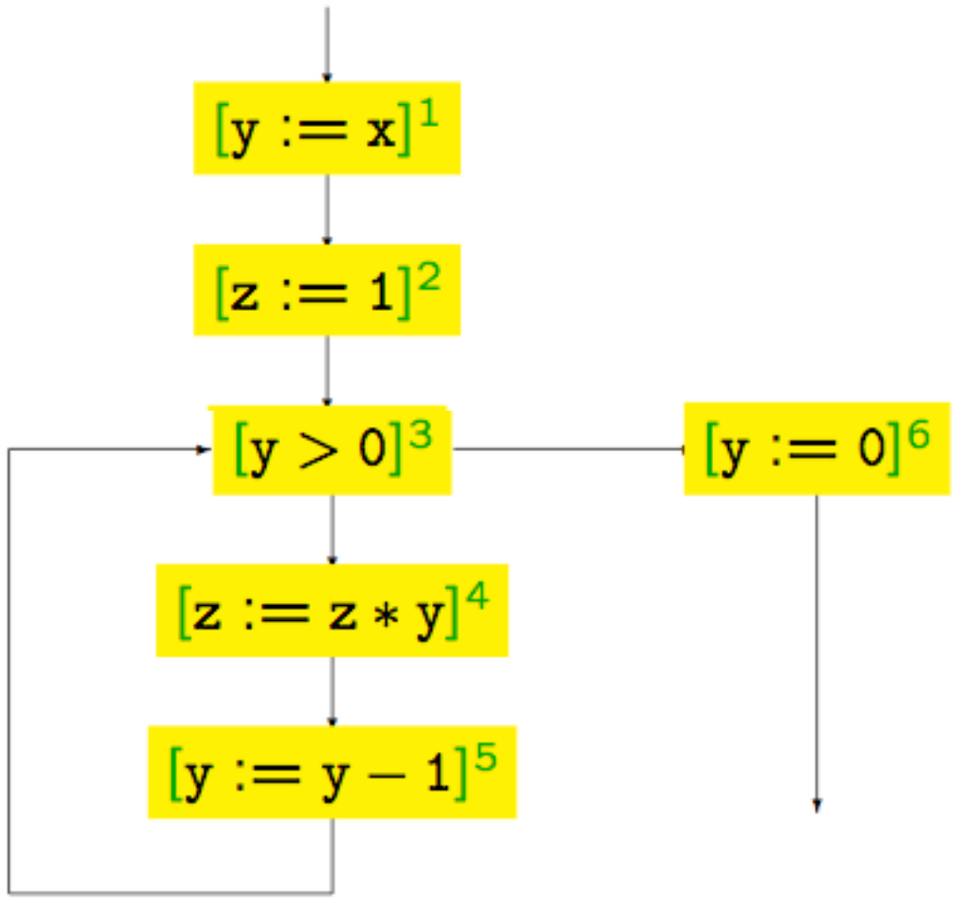
interesuje nas największe rozwiązanie

# Żywe zmienne

W każdym punkcie sterowania oblicz zbiór zmiennych żywych (zmienna może być w przyszłości użyta zanim zostanie przedefiniowana) na wyjściu z tego punktu.

analiza „wsteczna”

# Żywe zmienne




$$x_i^{\text{entry}} = x_i^{\text{exit}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

$$x_i^{\text{exit}} = \bigcup_{i \rightsquigarrow j} x_j^{\text{entry}}$$

$$x_6^{\text{exit}} = \emptyset$$

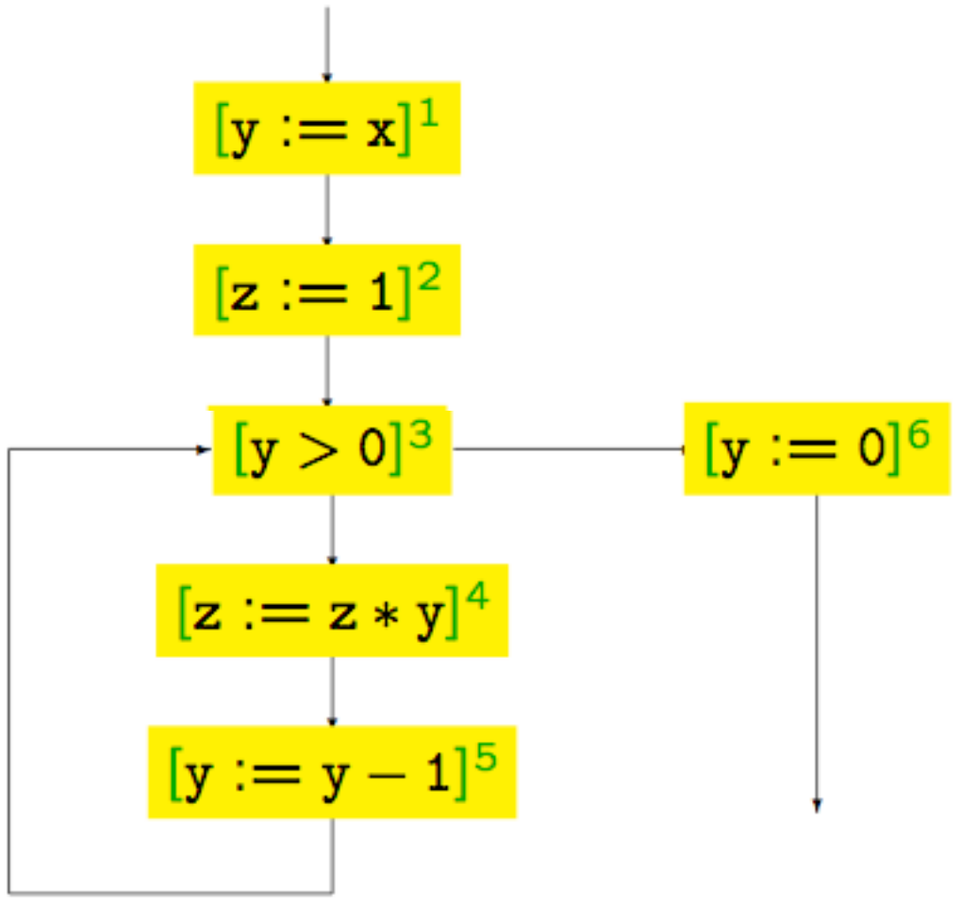
interesuje nas najmniejsze rozwiązanie

# Niezbędne wyrażenia

W każdym punkcie sterowania oblicz zbiór wyrażeń, które na pewno zostaną obliczone w przyszłości zanim któraś zmienna zostanie przedefiniowana.

analiza „wsteczna”

# Niezbędne wyrażenia




$$x_i^{\text{entry}} = x_i^{\text{exit}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

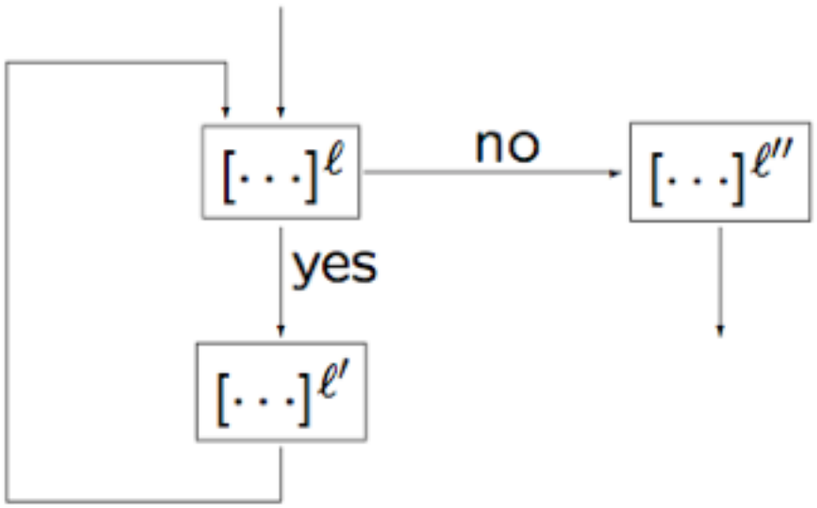
$$x_i^{\text{entry}} = \bigcap_{i \rightsquigarrow j} x_j^{\text{entry}}$$

$$x_6^{\text{exit}} = \emptyset$$

interesuje nas największe rozwiązanie

# Niezbędne wyrażenia

(while [x>1]$^{\ell}$ do [skip]$^{\ell'}$); [x:=x+1]$^{\ell''}$




$$x_i^{\text{entry}} = x_i^{\text{exit}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcap_{i \rightsquigarrow j} x_j^{\text{entry}}$$

dwa rozwiązania:

$$x_l^{\text{exit}} = \{x+1\},\ \emptyset$$

$$x_6^{\text{exit}} = \emptyset$$

interesuje nas największe rozwiązanie

# Interpretacja abstrakcyjna

# Ogólnie

$L$ - przestrzeń abstrakcyjna

$(L, \sqsubseteq)$ - krata zupełna

$\sqcup, \sqcap$ - kresy

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$x_i^{\mathrm{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}} \quad \sqcup \quad f_{\mathrm{init}}(x)$$

$$x_i^{\mathrm{exit}} = f(x)(x_i^{\mathrm{entry}})$$

# Ogólnie

interpretacja abtrakcyjna

$L$ - przestrzeń abstrakcyjna

$(L, \sqsubseteq)$ - krata zupełna

$\sqcup, \sqcap$ - kresy

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$x_i^{\mathrm{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}} \quad \sqcup \quad f_{\mathrm{init}}(x)$$

$$x_i^{\mathrm{exit}} = f(x)(x_i^{\mathrm{entry}})$$

# Analiza przepływu danych

- „docierające” przypisania
- dostępne wyrażenia
- żywe zmienne
- niezbędne wyrażenia
- propagacja stałych
- ...

# Analiza przepływu danych

- „docierające” przypisania $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- dostępne wyrażenia
- żywe zmienne
- niezbędne wyrażenia
- propagacja stałych
- ...

# Analiza przepływu danych

- „docierające” przypisania $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- dostępne wyrażenia $\mathcal{P}(\mathrm{Expr})$
- żywe zmienne
- niezbędne wyrażenia
- propagacja stałych
- ...

# Analiza przepływu danych

- „docierające” przypisania $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- dostępne wyrażenia $\mathcal{P}(\mathrm{Expr})$
- żywe zmienne $\mathcal{P}(\mathrm{Var})$
- niezbędne wyrażenia
- propagacja stałych
- ...

# Analiza przepływu danych

- „docierające” przypisania $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- dostępne wyrażenia $\mathcal{P}(\mathrm{Expr})$
- żywe zmienne $\mathcal{P}(\mathrm{Var})$
- niezbędne wyrażenia $\mathcal{P}(\mathrm{Expr})$
- propagacja stałych
- ...

# Analiza przepływu danych

- „docierające” przypisania $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- dostępne wyrażenia $\mathcal{P}(\mathrm{Expr})$
- żywe zmienne $\mathcal{P}(\mathrm{Var})$
- niezbędne wyrażenia $\mathcal{P}(\mathrm{Expr})$
- propagacja stałych $\mathrm{Var} \to \mathbb{Z}^\top$
- ...

# Rozdzielność

$$f(s)(l_1 \sqcup l_2) \quad = \quad f(s)(l_1) \ \sqcup \ f(s)(l_2)$$

zachodzi zawsze, jeśli:

$$L = \mathcal{P}(\mathcal{D}) \quad \mathcal{D} - \text{ skończony}$$

$$f(s)(l) = l \setminus l_1 \cup l_2$$

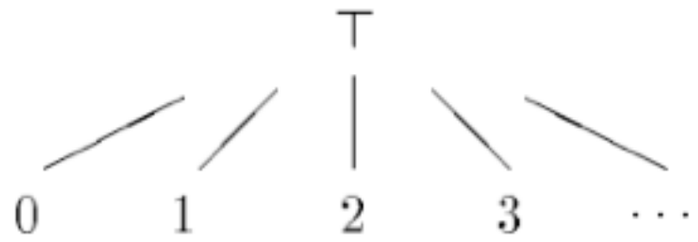
może nie zachodzić

# Propagacja stałych

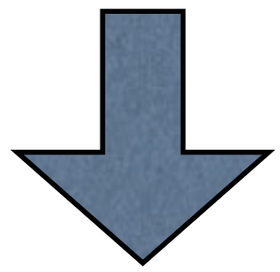
W każdym punkcie sterowania oblicz zbiór zmiennych, które mają stałą wartość niezależną od dotychczasowej ścieżki.

# Propagacja stałych

$$L = \mathrm{Var} \to \mathbb{Z}^{\top}$$




$[\mathtt{x:=6}]^1; [\mathtt{y:=3}]^2; \mathtt{while}\ [\mathtt{x} > \mathtt{y}]^3\ \mathtt{do}\ ([\mathtt{x:=x} - 1]^4; [\mathtt{z:=y} * \mathtt{y}]^6)$



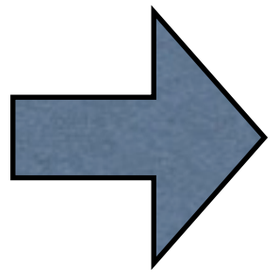
$[\mathtt{x:=6}]^1; [\mathtt{y:=3}]^2; \mathtt{while}\ [\mathtt{x} > 3]^3\ \mathtt{do}\ ([\mathtt{x:=x} - 1]^4; [\mathtt{z:=9}]^6)$

$$l_1(\mathrm{y}) = 5 \qquad l_2(\mathrm{y}) = -5$$
$$f(6)(l_1 \sqcup l_2)(\mathbf{z}) = \top$$
$$f(6)(l_1)(\mathbf{z}) = f(6)(l_2)(\mathbf{z}) = 25$$

# Algorytm

$$x_i^{\text{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\text{exit}} \quad \sqcup \quad f_{\text{init}}(x)$$

$$x_i^{\text{exit}} = f(x)(x_i^{\text{entry}})$$



$$\vec{x} = \vec{f}(\vec{x})$$

krata zupełna $L^{S \times \{\text{entry},\text{exit}\}}$ z porządkiem po współrzędnych

najmniejszy punkt stały funkcji monotonicznej $\vec{f}$

algorytm iteracyjny

zakładamy, że L ma tylko skończone łańcuchy

# LFP

$L$ - przestrzeń abstrakcyjna

$(L, \sqsubseteq)$ - krata zupełna

$\sqcup, \sqcap$ - kresy

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$x_i^{\mathrm{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}} \quad \sqcup \quad f_{\mathrm{init}}(x)$$

$$x_i^{\mathrm{exit}} = f(x)(x_i^{\mathrm{entry}})$$

# MOP

$L$ - przestrzeń abstrakcyjna

$(L, \sqsubseteq)$ - krata zupełna

$\sqcup, \sqcap$ - kresy

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$y_i^{\mathrm{entry}} = \bigsqcup \{ f(p) \mid p \in \mathrm{paths}^{\mathrm{entry}}(i) \}$$

$$y_i^{\mathrm{exit}} = \bigsqcup \{ f(p) \mid p \in \mathrm{paths}^{\mathrm{exit}}(i) \}$$

MOP $\sqsubseteq$ LFP

$$\vec{y} \sqsubseteq \vec{x}$$

# MOP $\sqsubseteq$ LFP

nie zawsze obliczalny

$$\vec{y} \sqsubseteq \vec{x}$$

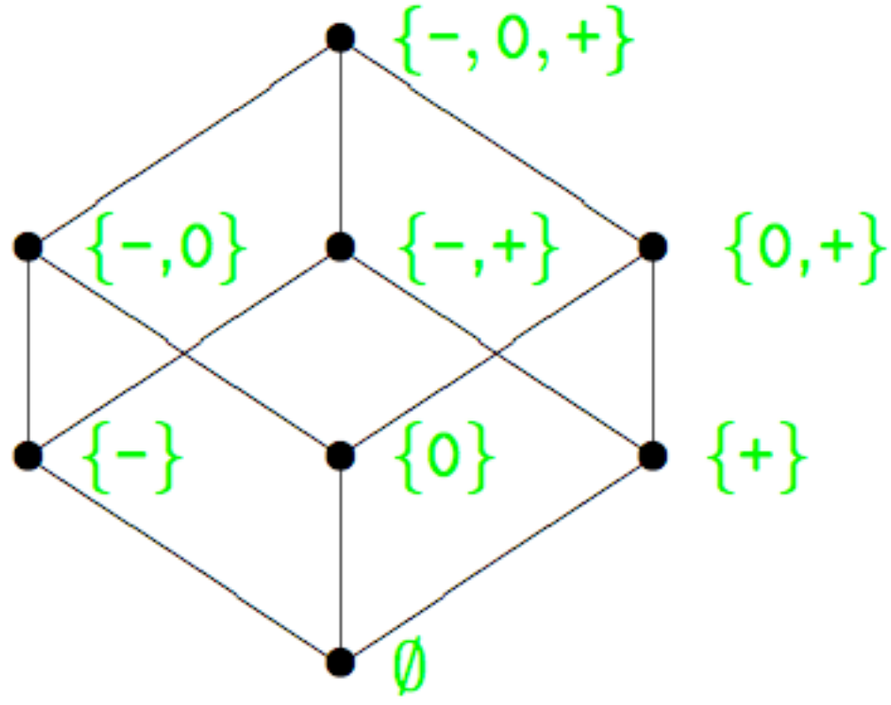
MOP = LFP

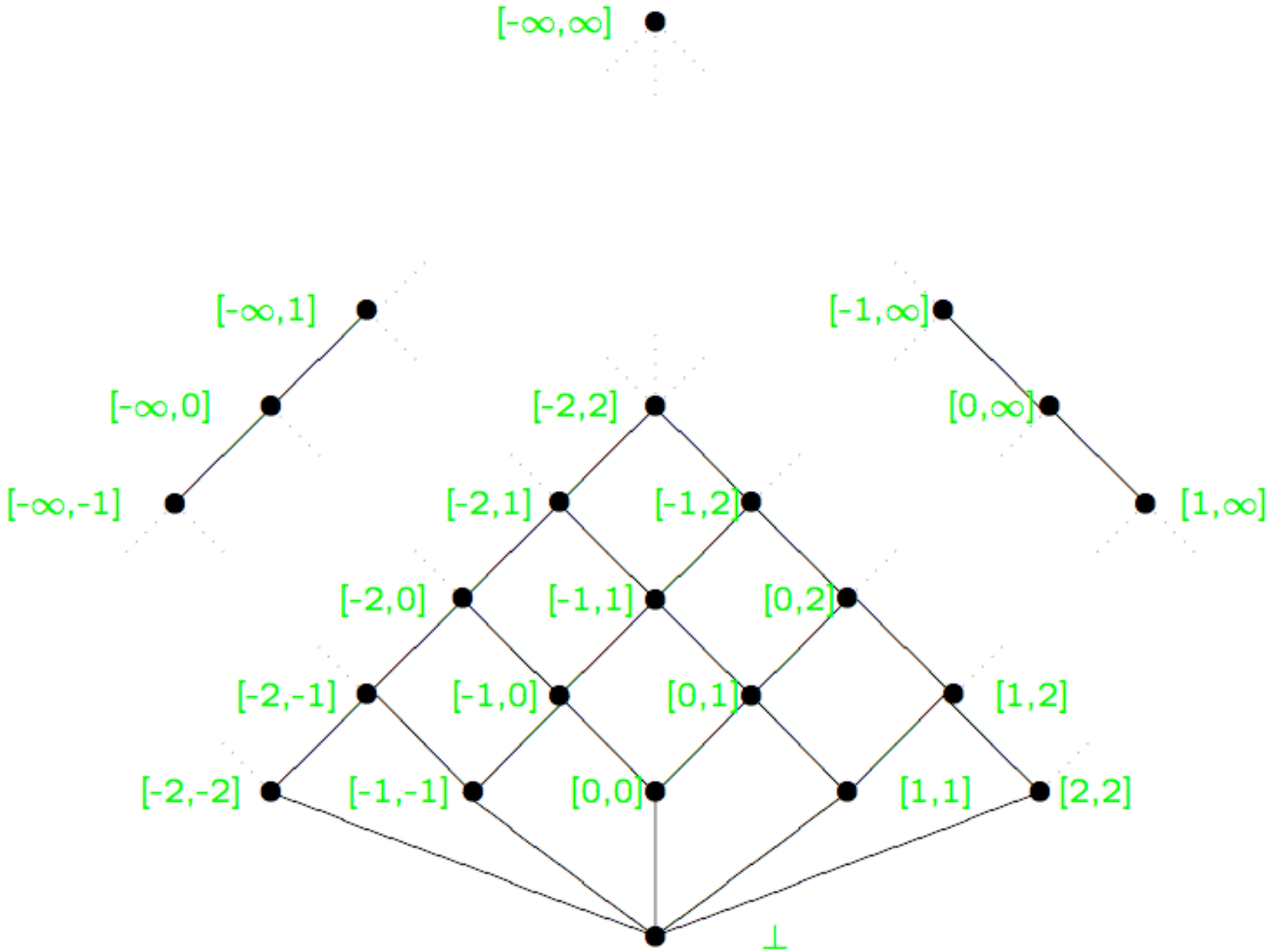
$$\vec{y} = \vec{x}$$

gdy zachodzi rozdzielność

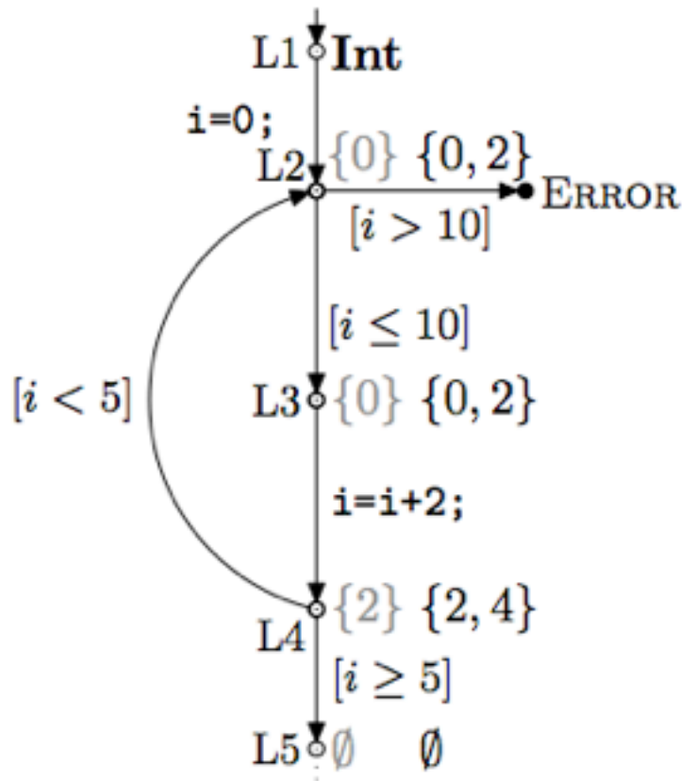
# Dziedziny nie-relacyjne

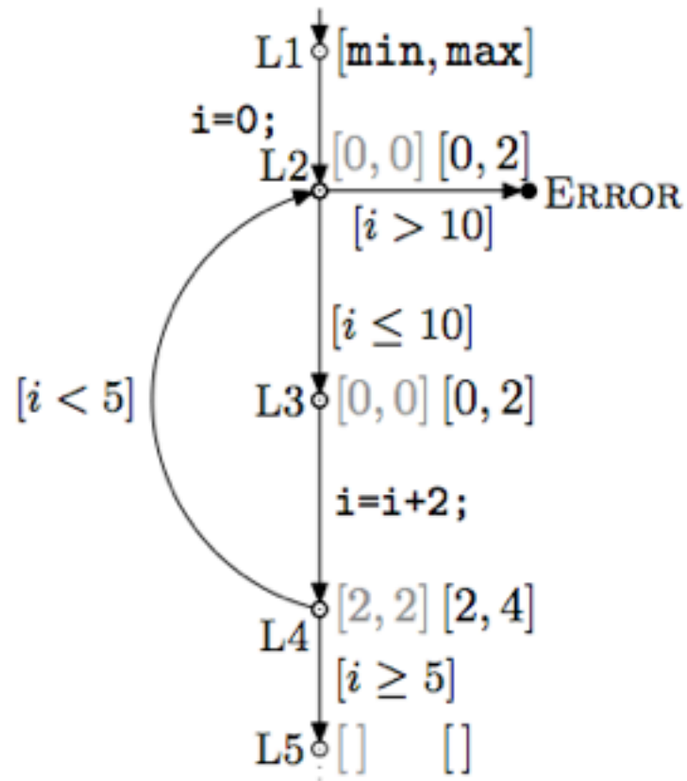
- znaki $\mathcal{P}(-, 0, +)$
- przedzialy $[n, m]$
- parzystość
- kongruencja modulo *k*

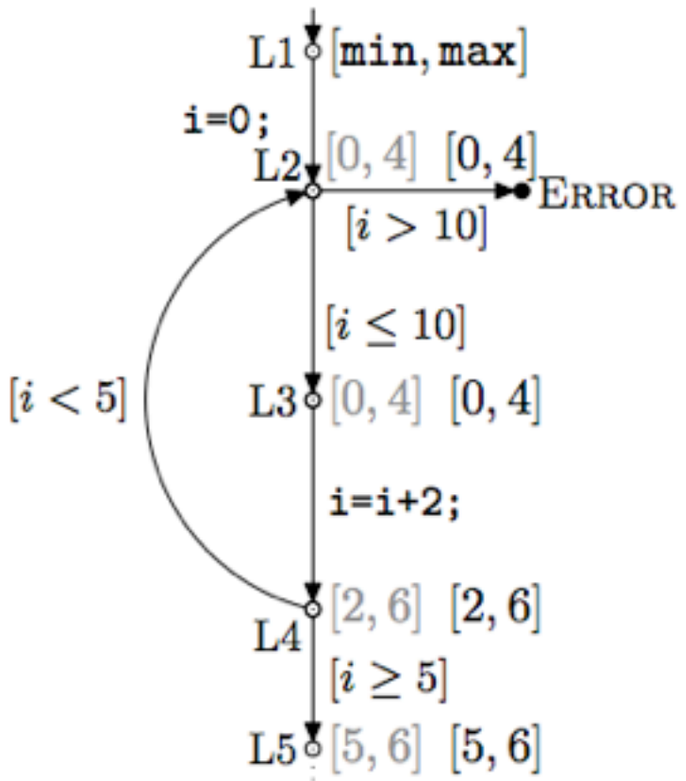



[-∞,∞]
[-∞,1]
[-∞,0]
[-∞,-1]
[-2,2]
[-2,1]
[-1,2]
[-1,∞]
[0,∞]
[1,∞]
[-2,0]
[-1,1]
[0,2]
[-2,-1]
[-1,0]
[0,1]
[1,2]
[-2,-2]
[-1,-1]
[0,0]
[1,1]
[2,2]
⊥

```
int i = 0;
do {
   assert(i <= 10);
   i = i+2;
} while (i < 5);
```



# Dziedziny relacyjne

precyzja ↓

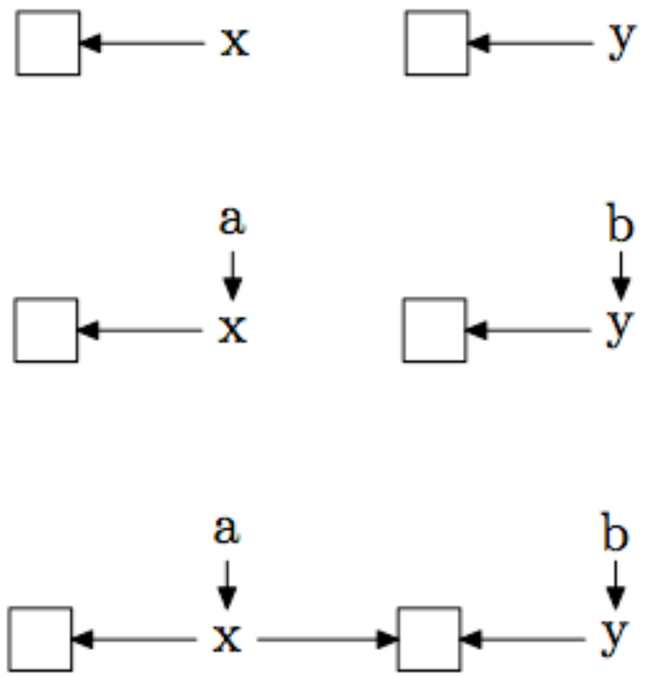
- DBM (macierze ograniczeń różnic) $x - y \leq c$
- ośmiokąty $\pm x \pm y \leq c$
- ośmiościany $\pm x_1 \ldots \pm x_n \leq c$
- wielościany $a_1 x_1 + \ \ldots \ + a_n x_n \leq c$

# Analiza kształtu

- grafy

# Analiza kształtu

```
int **a, **b, *x, *y;
x = (int*) malloc(sizeof(int));
y = (int*) malloc(sizeof(int));
a = &x;
b = &y;
*a = y;
```




a i b nie wskazują tej samej lokacji

x i y mogą wskazywać tę samą lokację

# Poprawność ?