# Programowanie w Logice

## Proste ograniczenia

Przemysław Kobylański

# Proste ograniczenia

## Ograniczenia arytmetyczne

Dwa wyrażenia arytmetyczne mogą być porównane następującymi relacjami:

```
Wyr1 #=  Wyr2
Wyr1 #\= Wyr2
Wyr1 #>= Wyr2
Wyr1 #=< Wyr2
Wyr1 #>  Wyr2
Wyr1 #<  Wyr2
```

# Proste ograniczenia

## Ograniczenia arytmetyczne

Wyrażenia `Wyr1` i `Wyr2` mogą być następujących postaci:

```
integer
variable
-Wyr
Wyr + Wyr
Wyr * Wyr
Wyr - Wyr
Wyr ^ Wyr
min(Wyr, Wyr)
max(Wyr, Wyr)
Wyr mod Wyr          -10 mod 3 = 2
Wyr rem Wyr          -10 rem 3 = -1
abs(Wyr)
Wyr // Wyr           -10 //  3 = trunc(-3.3333) = -3
Wyr div Wyr          -10 div 3 = floor(-3.3333) = -4
```

# Proste ograniczenia

## Ograniczenia arytmetyczne

### Example (Optymalne cięcie desek)

- Mamy do dyspozycji `N` desek długości 7 metrów, z których możemy wycinać na trzy sposoby:
  - pierwszy: dwa kawałki trzymetrowe i jeden kawałek jednometrowy
  - drugi: dwa kawałki dwumetrowe i jeden kawałek trzymetrowy
  - trzeci: cztery kawałki jednometrowe i jeden kawałek trzymetrowy
- Chcemy wyciąć `N1` kawałków jednometrowych, `N2` kawałków dwumetrowych i `N3` kawałków trzymetrowych.
- Każdy niepotrzebnie wycięty kawałek traktujemy jak zbędny odpad.
- **Jak ciąć deski aby zminimalizować odpad?**

## Proste ograniczenia
### Ograniczenia arytmetyczne

**Example (Optymalne cięcie desek cd.)**

Możliwe sposoby cięcia:


## Proste ograniczenia
### Ograniczenia arytmetyczne

**Example (Optymalne cięcie desek cd.)**

```
deski(N, N1, N2, N3, Sposoby, Odpad) :-
    Sposoby = [S1, S2, S3],
    Sposoby ins 0..N,
    S1 + S2 + S3 #=< N,
    W1 #= S1 + 4*S3,
    W2 #= 2*S2,
    W3 #= 2*S1 + S2 + S3,
    W1 #>= N1, W2 #>= N2, W3 #>= N3,
    Odpad #= (W1 - N1) + 2*(W2 - N2) + 3*(W3 - N3),
    once(labeling([min(Odpad)], Sposoby)).
```

## Proste ograniczenia
### Ograniczenia arytmetyczne

**Example (Optymalne cięcie desek cd.)**

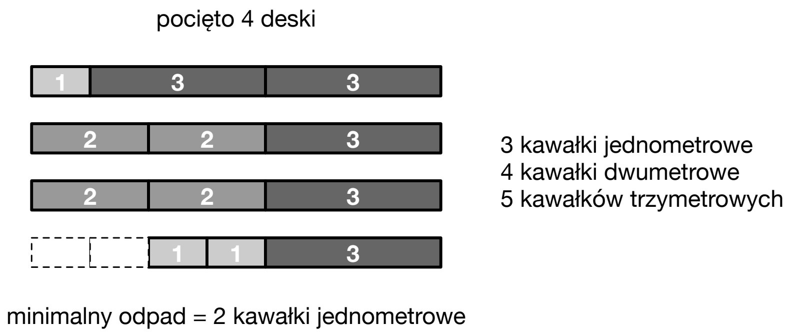
Przykłady zapytań:

```
?- deski(3, 3, 4, 5, X, Y).
false.

?- deski(4, 3, 4, 5, X, Y).
X = [1, 2, 1],
Y = 2.

?- deski(5, 3, 4, 5, X, Y).
X = [1, 2, 1],
Y = 2.
```

## Proste ograniczenia
### Ograniczenia arytmetyczne

**Example (Optymalne cięcie desek cd.)**

Optymalne rozwiązanie:



### Example (Optymalne cięcie desek cd.)

Modyfikacja modelu:

```
deski(N, N1, N2, N3, Sposoby, Odpad) :-
    Sposoby = [S1, S2, S3],
    Sposoby ins 0..N,
    Deski #= S1 + S2 + S3,
    Deski #=< N,
    W1 #= S1 + 4*S3,
    W2 #= 2*S2,
    W3 #= 2*S1 + S2 + S3,
    W1 #>= N1, W2 #>= N2, W3 #>= N3,
    Odpad #= (W1 - N1) + 2*(W2 - N2) + 3*(W3 - N3),
    once(labeling([min(Deski)], Sposoby)).
```



W ograniczeniach arytmetycznych możliwe są również następujące operacje bitowe:

```
\ Int               negacja
Int1 /\ Int2        koniunkcja
Int1 \/ Int2        alternatywa
Int1 >> Int2        logiczne przesunięcie w prawo
Int1 << Int2        arytmetyczne przesunięcie w lewo
lsb(Int)            pozycja najmniej znaczącej jedynki
msb(Int)            pozycja najbardziej znaczącej jedynki
popcount(Int)       liczba jedynek
Int1 xor Int2       alternatywa wykluczająca
```



### Example (Operacje bitowe)

```
?- X #= \ 5.              ?- X #= 5 /\ 3.
X = -6.                   X = 1.

?- X #= 5 \/ 3.           ?- X #= 5 xor 3.
X = 7.                    X = 6.

?- X #= 5 << 2.           ?- X #= 5 >> 2.
X = 20.                   X = 1.

?- X #= lsb(6).           ?- X #= msb(6).
X = 1.                    X = 2.

?- X #= popcount(6).
X = 2.
```



### Example (Dwa kryteria optymalizacji)

Jaka liczba z zakresu od 3000 do 4000 ma jak najbardziej odległe skrajne jedynki w swojej binarnej reprezentacji i liczba tych jedynek jest jak najmniejsza:

```
?- X in 3000..4000,
   Width #= msb(X)-lsb(X),
   Pop #= popcount(X),
   labeling([max(Width), min(Pop)], [X]).
X = 3073,
Width = 11,
Pop = 3 .
```

$$3073 = (110000000001)_2$$

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

Niech `X` i `Y` będą dwiema zmiennymi zero-jedynkowymi, których wartości będziemy interpretować jako prawda i fałsz. Spójniki logiczne możemy modelować w następujący sposób:

| **spójnik** | **liniowe ograniczenie** |
|---|---|
| $X \wedge Y$ | `X + Y #= 2` |
| $X \vee Y$ | `X + Y #>= 1` |
| $X \rightarrow Y$ | `Y #>= X` |
| $X \leftrightarrow Y$ | `X #= Y` |

**Example**

Implikacja $X \rightarrow Y$ jest równoważna alternatywie $\neg X \vee Y$.
Modelem dla tej alternatywy jest nierówność $1 - X + Y \geq 1$, która jest równoważna $Y \geq X$.

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

**Problem (Alternatywa ograniczeń)**

*Chcemy aby spełniona była alternatywa nierówności:*

$$A_1 \leq B_1 \vee A_2 \leq B_2.$$

*Z każdą z nierówności zwiążemy zmienną zero-jedynkową:*

$$\begin{aligned} X_1 = 1 &\rightarrow A_1 \leq B_1, \\ X_2 = 1 &\rightarrow A_2 \leq B_2. \end{aligned}$$

*Powyższe implikacje wyrazimy dwiema nierównościami:*

$$\begin{aligned} A_1 &\leq B_1 + M \cdot (1 - X_1), \\ A_2 &\leq B_2 + M \cdot (1 - X_2), \end{aligned}$$

*gdzie $M$ jest dostatecznie dużą liczbą całkowitą.*

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

**Problem (Alternatywa ograniczeń cd.)**

*Prawdziwość alternatywy $X_1 = 1 \vee X_2 = 1$ zapewni nam nierówność:*

$$X_1 + X_2 \geq 1.$$

*Ostatecznie program w Prologu modelujący alternatywę dwóch nierówności może wyglądać następująco:*

```
[X1, X2] ins 0..1,
A1 #=< B1 + M*(1 - X1),
A2 #=< B2 + M*(1 - X2),
X1 + X2 #>= 1
```

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

**Problem (Implikacja ograniczeń)**

*Chcemy aby spełniona była implikacja nierówności:*

$$A_1 \leq B_1 \rightarrow A_2 \leq B_2.$$

*Z każdą z nierówności zwiążemy zmienną zero-jedynkową:*

$$\begin{aligned} A_1 \leq B_1 &\rightarrow X_1 = 1, (\equiv X_1 = 0 \rightarrow A_1 > B_1) \\ X_2 = 1 &\rightarrow A_2 \leq B_2. \end{aligned}$$

*Powyższe implikacje wyrazimy dwiema nierównościami:*

$$\begin{aligned} A_1 &> B_1 - M \cdot X_1, \\ A_2 &\leq B_2 + M \cdot (1 - X_2), \end{aligned}$$

*gdzie $M$ jest dostatecznie dużą liczbą całkowitą.*

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

**Problem (Implikacja ograniczeń cd.)**

*Prawdziwość implikacji $X_1 = 1 \to X_2 = 1$ zapewni nam nierówność:*

$$X_2 \geq X_1.$$

*Ostatecznie program w Prologu modelujący implikację dwóch nierówności może wyglądać następująco:*

```
[X1, X2] ins 0..1,
A1 #>  B1 - M*X1,
A2 #=< B2 + M*(1 - X2),
X2 #>= X1
```

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

**Example (Model implikacji)**

Wyrazimy implikację $A_1 \leq B_1 \to A_2 \leq B_2$:

```
?- [A1, A2, B1, B2] ins 0..100, [X1, X2] ins 0..1,
   A1 #> B1 - 200*X1, A2 #=< B2 + 200*(1-X2), X2 #>= X1.
A1 in 0..100,
B1+1#=<A1+200*X1,
B1 in 0..100,
X1 in 0..1,
X2#>=X1,
X2 in 0..1,
A2+200*X2#=<B2+200,
A2 in 0..100,
B2 in 0..100.
```

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

**Example (Model implikacji cd.)**

Prawdziwy poprzednik:

```
?- [A1, A2, B1, B2] ins 0..100, [X1, X2] ins 0..1,
   A1 #> B1 - 200*X1, A2 #=< B2 + 200*(1-X2), X2 #>= X1,
   A1 #=< 20, B1 #>= 30.
X1 = X2, X2 = 1,
A1 in 0..20,
B1 in 30..100,
A2 in 0..100,
A2#=<B2,               % następnik też musi być prawdziwy
B2 in 0..100.
```

## Proste ograniczenia
### Modelowanie spójników logicznych liniowymi ograniczeniami

**Example (Model implikacji cd.)**

Fałszywy następnik:

```
?- [A1, A2, B1, B2] ins 0..100, [X1, X2] ins 0..1,
   A1 #> B1 - 200*X1, A2 #=< B2 + 200*(1-X2), X2 #>= X1,
   B2 #=< 20, A2 #>= 30.
X1 = X2, X2 = 0,
A1 in 1..100,
B1+1#=<A1,             % poprzednik też musi być fałszywy
B1 in 0..99,
A2 in 30..100,
B2 in 0..20.
```

## Proste ograniczenia
### Reifikacja ograniczeń

- Z każdym możliwym ograniczeniem arytmetycznym można związać zmienną zero-jedynkową przy użyciu relacji #<==>/2, w ten sposób, że ograniczenie zachodzi wtedy i tylko wtedy, gdy zmienna przyjmuje wartość 1.
- Niech `A` i `B` będą dowolnymi wyrażeniami arytmetycznymi a `X` zmienną zero-jedynkową. Możliwe reifikacje ograniczeń:

```
(A #=  B) #<==> X
(A #\= B) #<==> X
(A #>= B) #<==> X
(A #=< B) #<==> X
(A #>  B) #<==> X
(A #<  B) #<==> X
```

## Proste ograniczenia
### Reifikacja ograniczeń

**Example**

Chcemy aby z trzech równań $A_1 = B_1, A_2 = B_2, A_3 = B_3$ zachodziły dokładnie dwa:

```
(A1 #= B1) #<==> X1,
(A2 #= B2) #<==> X2,
(A3 #= B3) #<==> X3,
X1 + X2 + X3 #= 2
```

## Proste ograniczenia
### Spójniki logiczne między ograniczeniami

Dla ograniczeń arytmetycznych `P` i `Q` możemy budować bardziej złożone ograniczenia:

```
#\ P          prawdziwe gdy P fałszywe
P #/\ Q       prawdziwe gdy P i Q prawdziwe
P #\/ Q       prawdziwe gdy P lub Q prawdziwe
P #\ Q        prawdziwe gdy P albo Q prawdziwe
P #==> Q      prawdziwe gdy P implikuje Q
P #<== Q      prawdziwe gdy Q implikuje P
P #<==> Q     prawdziwe gdy P i Q są równoważne
```

## Proste ograniczenia
### Spójniki logiczne między ograniczeniami

**Example (Alternatywa ograniczeń)**

```
?- (A1 #=< B1) #\/ (A2 #=< B2).
B1#>=A1#<==>_1,
_1 in 0..1,
_1#\/_2#<==>1,
_2 in 0..1,
B2#>=A2#<==>_2.
```

### Example (Implikacja ograniczeń)

```
?- (A1 #=< B1) #==> (A2 #=< B2).
B1#>=A1#<==>_1,
_1 in 0..1,
_1#==>_2,
_2 in 0..1,
B2#>=A2#<==>_2.
```

### Example (Równoważność ograniczeń)

```
?- (A1 #=< B1) #<==> (A2 #=< B2).
B1#>=A1#<==>_1,
_1 in 0..1,
B2#>=A2#<==>_1.
```