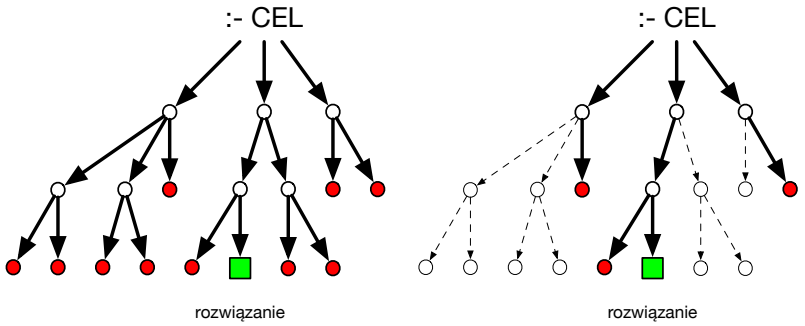
Programowanie w Logice

Zmienne, dziedziny i ograniczenia

Przemysław Kobylański

Zmienne, dziedziny i ograniczenia

Idea programowania ograniczeń (więzów)



a) Prolog bez więzów

b) Prolog z więzami

Zmienne, dziedziny i ograniczenia

Idea programowania ograniczeń (więzów)

Example (SEND+MORE=MONEY)

Klasyczna krypto-arytmetyczna łamigłówka:

$$\begin{array}{r} \text{SEND} \\ +\ \text{MORE} \\ \hline \text{MONEY} \end{array}$$

1. Cyfry S, E, N, D, M, O, R, Y są parami różne.
2. Liczby nie rozpoczynają się zerem.

Jakie cyfry znajdują się na poszczególnych pozycjach aby operacja dodawania była poprawna?

Zmienne, dziedziny i ograniczenia

Idea programowania ograniczeń (więzów)

Example (SEND+MORE=MONEY cd.)

```
% Prolog bez więzów:
digit(0). digit(1). digit(2). digit(3). digit(4).
digit(5). digit(6). digit(7). digit(8). digit(9).

solve1([[S,E,N,D],[M,O,R,E],[M,O,N,E,Y]]) :-
    digit(S), digit(E), digit(N), digit(D), % generowanie
    digit(M), digit(O), digit(R), digit(Y), % wartości
    S =\= E, S =\= N, S =\= D, S =\= M, S =\= O, S =\= R,
    S =\= Y, E =\= N, E =\= D, E =\= M, E =\= O, E =\= R,
    E =\= Y, N =\= D, N =\= M, N =\= O, N =\= R, N =\= Y,
    D =\= M, D =\= O, D =\= R, D =\= Y, M =\= O, M =\= R,
    M =\= Y, O =\= R, O =\= Y, R =\= Y, S > 0,   M > 0,
    1000*S + 100*E + 10*N + D + 1000*M + 100*O + 10*R + E
    =:=
    10000*M + 1000*O + 100*N + 10*E + Y.
```

Example (SEND+MORE=MONEY cd.)

```
% Prolog z więzami:
solve2([[S,E,N,D],[M,O,R,E],[M,O,N,E,Y]]) :-
    Vars = [S, E, N, D, M, O, R, Y],
    Vars ins 0..9,
    all_different(Vars),
    S #> 0, M #> 0,
    1000*S + 100*E + 10*N + D + 1000*M + 100*O + 10*R + E
    #=
    10000*M + 1000*O + 100*N + 10*E + Y,
    label(Vars). % generowanie rozwiązań
```


Example (SEND+MORE=MONEY cd.)

- Bez użycia więzów jest $10^8 = 100,000,000$ potencjalnych rozwiązań.
- Dzięki użyciu więzów liczba potencjalnych rozwiązań **przed** ich generowaniem zostaje zredukowana do $4 \times 4 \times 7 \times 7 \times 7 = 5,488$ (ponad 18 tysięcy razy mniej):

	9	4..7	5..8	2..8
+	1	0	2..8	4..7
1	0	5..8	4..7	2..8

Example (SEND+MORE=MONEY cd.)

Porównanie czasu działania:

```
?- time(solve1(X)).
% 878,558,137 inferences, 74.980 CPU in 74.982 seconds (10(
X = [[9, 5, 6, 7], [1, 0, 8, 5], [1, 0, 6, 5, 2]] ;
% 33,939,529 inferences, 2.910 CPU in 2.910 seconds (100% (
false.

?- time(solve2(X)).
% 7,380 inferences, 0.003 CPU in 0.003 seconds (100% CPU, 2
X = [[9, 5, 6, 7], [1, 0, 8, 5], [1, 0, 6, 5, 2]] ;
% 1,999 inferences, 0.001 CPU in 0.001 seconds (100% CPU, 2
false.
```


Example (SEND+MORE=MONEY cd.)

	`solve1/1`	`solve2/1`	Przyspieszenie
Pierwsze rozwiązanie	74.980s	0.003s	25 tys. razy
Wszystkie rozwiązania	77.890s	0.004s	19 tys. razy

- Podczas działania predykatu `label/1` nie odbywa się **pełen** przegląd pozostałych potencjalnych rozwiązań.
- Po każdym podstawieniu wartości za zmienną ponownie propagowane są narzucone ograniczenia i możliwe jest dalsze zawężanie dziedzin.
- Kolejna wartość zmiennej wybierana jest z aktualnie zawężonej dziedziny.
- Przyspieszenie może być **większe** niż tylko wynikające z oszacowania liczby potencjalnych rozwiązań!

Zmienne, dziedziny i ograniczenia
Atrybuty zmiennych

- **Programowanie ograniczeń** (ang. *constraint programming*) opiera się na związaniu ze zmiennymi zbiorów dopuszczalnych wartości, nazywanymi dziedzinami.
- Jeśli któraś ze zmiennych ma pustą dziedzinę, to wykrywana jest sprzeczność i następuję nawrót.
- Dziedziny zmiennych zawężane są przez uwzględnianie (propagowanie) ograniczeń wiążących wartości zmiennych.
- Aby możliwe było związanie ze zmienną dziedziny jej wartości należy użyć atrybutów zmiennych.
- Atrybuty prologowych zmiennych zaproponował Christian Holzbaur w roku 1992.

Zmienne, dziedziny i ograniczenia
Atrybuty zmiennych

- `attvar(?Term)` Sprawdza czy argument jest zmienną z atrybutami.
- `put_attr(+Var, +Module, +Value)` Jeśli `Var` jest zmienną lub zmienną z atrybutami, to ustalana jest wartość `Value` atrybutu o nazwie `Module`. Jeśli wcześniej zmienna miała atrybut o tej nazwie, to stara wartość jest przechowana i w przypadku nawrotu będzie odtworzona. Jeśli `Var` nie jest zmienną, to zgłaszany jest wyjątek.
- `get_attr(+Var, +Module, -Value)` Pobierana jest wartość atrybutu o nazwie `Module` zmiennej `Var`. Jeśli `Var` nie jest zmienną z atrybutem lub nie ma ustalonej wartości atrybutu `Module`, to kończy się niepowodzeniem.
- `del_attr(+Var, +Module)` Usuwa atrybut `Module` zmiennej `Var`.

Zmienne, dziedziny i ograniczenia
Atrybuty zmiennych

- Dzięki predykatowi `attr_unify_hook/2` można odpowiednio obsłużyć zmianę atrybutów podczas unifikacji dwóch zmiennych z atrybutami.
- Obsługa atrybutu jest związana z modułem o tej samej nazwie co atrybut i predykat `attr_unify_hook/2` dla danego atrybutu powinien być zdefiniowany w odpowiednim module o tej samej nazwie, co nazwa atrybutu.
- Predykat `attr_unify_hook(AttValue, VarValue)` jest wywoływany **po** unifikacji zmiennej (z atrybutem o wartości `AttValue`) z termem `VarValue`.
- Nieterminal `attribute_goals(Var)` służy do generowania gramatyką metamorficzną postaci w jakiej pojawi się atrybut zmiennej `Var` w drukowanej odpowiedzi.

Zmienne, dziedziny i ograniczenia
Atrybuty zmiennych

Example (Korzystanie z atrybutów)

Przykład zaczerpnięty z **SWI Prolog Reference Manual**.

```
:- module(domain,
          [ domain/2                    % Var, ?Domain
          ]).
:- use_module(library(ordsets)).

domain(X, Dom) :-
    var(Dom), !,
    get_attr(X, domain, Dom).
domain(X, List) :-
    list_to_ord_set(List, Domain),
    put_attr(Y, domain, Domain),
    X = Y.
```

Zmienne, dziedziny i ograniczenia
Atrybuty zmiennych

Example (Korzystanie z atrybutów cd.)

```
attr_unify_hook(Domain, Y) :-
    (   get_attr(Y, domain, Dom2)
    ->  ord_intersection(Domain, Dom2, NewDomain),
        (   NewDomain == []
        ->  fail
        ;   NewDomain = [Value]
        ->  Y = Value
        ;   put_attr(Y, domain, NewDomain)
        )
    ;   var(Y)
    ->  put_attr( Y, domain, Domain )
    ;   ord_memberchk(Y, Domain)
    ).
```

Zmienne, dziedziny i ograniczenia
Atrybuty zmiennych

Example (Korzystanie z atrybutów cd.)

```
attribute_goals(X) -->
    { get_attr(X, domain, List) },
    [domain(X, List)].
```

Przykłady zapytań:

```
?- domain(X, [a, b]), X = c.
false.

?- domain(X, [a, b]), domain(X, [b, c]).
X = b.

?- domain(X, [a, b, c]), domain(X, [b, c, d]).
domain(X, [b, c]).
```

Zmienne, dziedziny i ograniczenia
Dziedziny

W SWI-Prologu dostępne są następujące moduły dostarczające więzów na różnych zbiorach wartości:

- clpfd skończone dziedziny liczb całkowitych
- clpb wartości boolowskie
- clpq liczby wymierne (ułamki w postaci ilorazu licznika przez mianownik)
- clpr liczby rzeczywiste zmiennopozycyjne

W części wykładów poświęconych programowaniu ograniczeń omówimy wybrane więzy z modułu **clpfd**.
Zakładamy, że programy źródłowe zawierają dyrektywę:

```
:- use_module(library(clpfd)).
```

Zmienne, dziedziny i ograniczenia
Dziedziny

- Jeśli nie podano dziedziny dla zmiennej, to przyjmuje się, że dziedziną jej wartości jest cały dostępny zakres liczb całkowitych `inf .. sup`, gdzie `inf` i `sup` to, odpowiednio, najmniejsza i największa dostępna liczba całkowita.
- Predykat `Var in Dom` definiuje dziedzinę `Dom` dla zmiennej `Var`.
- Dziedzina `Dom` może być zadana jako: pojedyncza liczba całkowita, zakres wartości `Lo..Hi` albo jako suma mnogościowa dwóch dziedzin `Dom1 \/ Dom2`.
  ```
  ?- X in 1..3 \/ 5..8, Y in 3..4 \/ 3..9, X = Y.
  X = Y,
  Y in 3\/5..8.
  ```
- Aktualną dziedzinę zmiennej można sprawdzić wywołując `fd_dom(Var, Dom)`.
- Dolny i górny kres aktualnej dziedziny można sprawdzić wywołując, odpowiednio, `fd_inf(Var, Lo)` i `fd_sup(Var, Hi)`.

Zmienne, dziedziny i ograniczenia

Dziedziny

- Jeśli wielu zmiennym na liście `Vars` chcemy określić tę samą dziedzinę `Dom`, to możemy to zrobić predykatem `Vars ins Dom`.

```
?- [Wiersz, Kolumna] ins 1..8.
Wiersz in 1..8,
Kolumna in 1..8.
```

- Dla przykładu, poniższy predykat `zmienne(N, L)`, dla danego całkowitego `N`, tworzy listę `N` zmiennych o dziedzinach `1..N`:

```
zmienne(N, L) :- length(L, N), L ins 1..N.

?- zmienne(5, X).
X = [_12418, _12424, _12430, _12436, _12442],
_12418 in 1..5,
_12424 in 1..5,
_12430 in 1..5,
_12436 in 1..5,
_12442 in 1..5.
```

Zmienne, dziedziny i ograniczenia

Ograniczenia

- Do najprostszych ograniczeń w module **clpfd** należą ograniczenia arytmetyczne.
- Jeśli `Wyr1` i `Wyr2` są dwoma wyrażeniami arytmetycznymi, to dostępne są następujące relacje:

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

Zmienne, dziedziny i ograniczenia

Ograniczenia

Example (Ciąg Fibonacciego)

Zdefiniujemy predykat `fib/1`, który znajduje wszystkie liczby w ciągu Fibonacciego:

```
fib(0).
fib(1).
fib(1).
fib(F) :-
    fib(1, 1, F).

fib(F1, F2, F) :-
    F #> F2,
    F3 #= F1+F2,
    (   F #= F3
    ;   fib(F2, F3, F)).
```

Zmienne, dziedziny i ograniczenia

Ograniczenia

Example (Ciąg Fibonacciego cd.)

Początkowe wyrazy:

```
?- fib(X).
X = 0 ;          X = 1 ;
X = 1 ;          X = 2 ;
X = 3 ;          X = 5 ;
X = 8 ;          X = 13 ;
...
```

Wyrazy mniejsze od 10:

```
?- X #< 10, fib(X).
X = 0 ;           X = 1 ;
X = 1 ;           X = 2 ;
X = 3 ;           X = 5 ;
X = 8 ;
false.
```

Example (Ciąg Fibonacciego cd.)

Wyrazy większe od 20:

```
?- X #> 20, fib(X).
X = 21 ;                X = 34 ;
X = 55 ;                X = 89 ;
X = 144 ;               X = 233 ;
...
```

Wyrazy z zakresu od 1000 do 5000:

```
?- X in 1000..5000, fib(X).
X = 1597 ;
X = 2584 ;
X = 4181 ;
false.
```


Example (Ciąg Fibonacciego cd.)

Wyrazy podzielne przez 13:

```
?- X mod 13 #= 0, fib(X).
X = 0 ;                 X = 13 ;
X = 377 ;               X = 10946 ;
X = 317811 ;            X = 9227465 ;
X = 267914296 ;         X = 7778742049 ;
...
```

Sprawdzenie czy dana wartość jest w ciągu Fibonacciego:

```
?- fib(39088169).
true .

?- fib(54165426).
false.
```


- Ograniczenia wiążą ze sobą zmienne.
- Niech $C(x_1, \ldots, x_n)$ będzie ograniczeniem na zmiennych $x_1, \ldots, x_n$.
- Niech $D_i = Dom(x_i)$, dla $i = 1, \ldots, n$, będzie dziedziną i-tej zmiennej.
- Jeśli istnieje wartość $a_i \in D_i$, taka że nie można w dziedzinach pozostałych zmiennych wybrać dla nich wartości $b_1 \in D_1, \ldots, b_{i-1} \in D_{i-1}, b_{i+1} \in D_{i+1}, \ldots, b_n \in D_n$, tak aby wybrane wartości i wartość a_i spełniały $C(b_1, \ldots, b_{i-1}, a_i, b_{i+1}, \ldots, b_n)$, to wartość a_i należy usunąć z dziedziny D_i przyjmując $D_i := D_i - \{a_i\}$.
- Taki proces zawężania dziedzin nazywa się propagowaniem ograniczenia.
- Powtarza się go dopóki zmieniają się dziedziny zmiennych.

Example (Propagowanie ograniczenia)

Niech $C(X, Y, Z) \equiv X + 2Y = Z$ oraz $D_X = D_Y = D_Z = 1..10$.

$\inf_X$	$\sup_X$	$\inf_Y$	$\sup_Y$	$\inf_Z$	$\sup_Z$	$\lvert D_X\rvert \cdot \lvert D_Y\rvert \cdot \lvert D_Z\rvert$
1	10	1	10	(1)	10	1000
1	(10)	1	10	3	10	800
1	8	1	(10)	3	10	640
1	8	1	4	3	10	256

```
?- [X, Y, Z] ins 1..10, X + 2*Y #= Z.
X in 1..8,
X+2*Y#=Z,
Y in 1..4,
Z in 3..10.
```

Zmienne, dziedziny i ograniczenia
Etykietowanie zmiennych

- W wyniku propagacji ograniczeń zostają zawężone dziedziny zmiennych.
- Jeśli któraś z dziedzin zawęzi się do zbioru pustego, to jest niepowodzenie i nawrót.
- Aby poznać rozwiązanie spełniające wszystkie ograniczenia, konieczne jest przypisanie każdej ze zmiennych wartości z jej dziedziny.

Zmienne, dziedziny i ograniczenia
Etykietowanie zmiennych

Example (Trzy z dwóch)

Rozpatrzmy następujący cel:

```
?- [X, Y, Z] ins 0..1, X #\= Y, X #\= Z, Y #\= Z.
```

Szukamy w nim trzech parami różnych wartości wybranych ze zbioru dwuelementowego $\{0, 1\}$. Chociaż jest to niemożliwe, to Prolog zwraca nam następującą odpowiedź:

```
X in 0..1,
X#\=Z,
X#\=Y,
Z in 0..1,
Y#\=Z,
Y in 0..1.
```

Zmienne, dziedziny i ograniczenia
Etykietowanie zmiennych

Example (Trzy z dwóch cd.)

- Dlaczego nie udało się Prologowi zawęzić dziedzin $D_X = D_Y = D_Z = \{0, 1\}$ do zbiorów pustych?
- Ze względu na każde z trzech ograniczeń, nie mógł on z dziedziny jednej zmiennej usunąć żadnej wartości, dlatego że w dziedzinie drugiej zmiennej znajdowała się wartość, która z tą pierwszą spełnia warunek „różne”.
- Na przykład, na podstawie ograniczenia `X #\= Y` nie mógł z D_X usunąć 0, bo wartość $1 \in D_Y$ spełnia $0 \neq 1$. Analogicznie nie mógł z D_X usunąć 1, bo wartość $0 \in D_Y$ spełnia $1 \neq 0$.
- Trzeba zmusić Prolog by postarał się podstawić pod wszystkie trzy zmienne takie wartości z ich dziedzin, by spełniony były wszystkie ograniczenia.

Zmienne, dziedziny i ograniczenia
Etykietowanie zmiennych

Example (Trzy z dwóch cd.)

Możemy to zrobić korzystając z predykatu `indomain/1`:

```
?- [X, Y, Z] ins 0..1, X #\= Y, X #\= Z, Y #\= Z,
   indomain(X), indomain(Y), indomain(Z).
false.
```

Dopiero teraz Prolog zauważa, że nie ma rozwiązania na tak zadane pytanie.
Jeśli chcemy wybrać wartości dla wszystkich zmiennych na danej liście, to możemy użyć predykatu `label/1`:

```
?- [X, Y, Z] ins 0..1, X #\= Y, X #\= Z, Y #\= Z,
   label([X, Y, Z]).
false.
```

Zmienne, dziedziny i ograniczenia

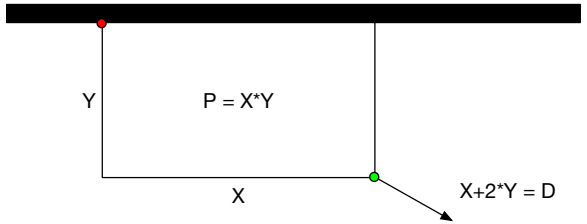
Etykietowanie zmiennych

- Poza predykatem `label/1` dostępny jest jeszcze predykat `labeling(Options, Vars)`, który możliwia sterowanie procesem etykietowania zmiennych.
- Jedną z możliwości jakie udostępnia `labeling/2` jest rozwiązywanie problemów optymalizacyjnych przez zadawanie funkcji celu, której wartość można maksymalizować opcją `max(Obj)` lub minimalizować opcją `min(Opt)`.
- Istnieje możliwość podania strategii wyboru kolejnej zmiennej z listy zmiennych oraz podania strategii wyboru kolejnej wartości z dziedziny.
- Predykat `labeling/2` zostanie dokładniej omówiony na kolejnych wykładach.

Zmienne, dziedziny i ograniczenia

Etykietowanie zmiennych

Example (Ogrodzenie)




```
ogrodzenie(D, X, Y, P) :-
    [X, Y] ins 0 .. D,
    X + 2*Y #= D, P #= X*Y,
    labeling([max(P)], [X, Y]).

?- ogrodzenie(100, X, Y, P).
X = 50, Y = 25, P = 1250 .
```