



Programowanie deklaratywne

Artur Michalski
Informatyka II rok



Plan wykładu

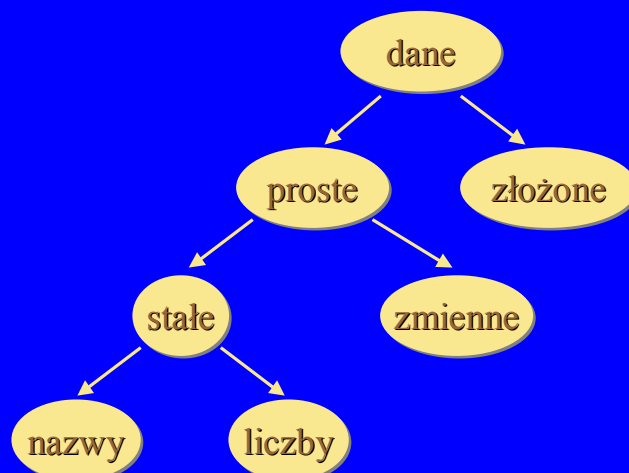
- Wprowadzenie do języka Prolog
- *Budowa składniowa* i interpretacja programów prologowych
- Listy, operatory i operacje arytmetyczne
- Sterowanie mechanizmem nawrotów
- Predefiniowane procedury prologowe
- Styl i technika programowania w Prologu

Budowa składniowa i znaczenie programów w Prologu

- Reprezentacja danych w Prologu
- Mechanizm uzgadniania (ang. *unification*)
- Związki języka Prolog z logiką formalną
- Podsumowanie

Reprezentacja danych w Prologu

Klasyfikacja rodzajów danych w Prologu



Reprezentacja danych w Prologu

- **Atomy** - składają się z:
 - dużych liter: A, B, ..., Z
 - małych liter: a, b, ..., z
 - cyfr: 0, 1, ..., 9
 - znaków specjalnych: +-*/<>=.&_~
- i są ciągami:
 - liter, cyfr i znaków podkreślenia (zaczynającymi się od małej litery): anna, x12, y_, _z, a_b
 - znaków specjalnych: <--->, ==>, ...
 - lub napisami: 'Ania', 'co to', 'kto_to'

Reprezentacja danych w Prologu

- **Zmienne** - ciągi liter, cyfr i znaków podkreślenia zaczynające się od dużej litery lub podkreślenia
- Zmienna anonimowa - tylko znak podkreślenia

Przykład

```
ma_dziecko(X) :- rodzic(X,Y) .
```

ponieważ konkluzja klauzuli nie zależy od wartości zmiennej **Y**, można ją zastąpić przez zmienną nie posiadającą identyfikatora tzw. *zmienną anonimową*:

```
ma_dziecko(X) :- rodzic(X,_) .
```

Reprezentacja danych w Prologu

Zmienna anonimowa c.d.

Każde wystąpienie zmiennej anonimowej w danej klauzuli reprezentuje inną zmienną

Przykład

```
ktos_ma_dziecko :- rodzic(_, _).
```

jest równoważne:

```
ktos_ma_dziecko :- rodzic(X, Y).
```

a nie:

```
ktos_ma_dziecko :- rodzic(X, X).
```

Reprezentacja danych w Prologu

Jeśli zmienna anonimowa zostanie użyta w pytaniu, jej wartość *nie zostanie* wyświetlona:

Przykład

Które osoby mają dzieci (bez wskazywania samych dzieci)?

Zapis w Prologu:

```
?- rodzic(X, _).
```

Odpowiedź:

```
X = torek;
```

```
...
```

Reprezentacja danych w Prologu

Zasięg leksykalny zmiennej (identyfikatora) ograniczony jest tylko do jednej klauzuli:

Przykład

```
ma_dziecko(X) :- rodzic(X, _).  
dziecko(X, Y) :- rodzic(Y, X).
```

zmienna **X** oznacza inną zmienną w każdej z klauzul, czyli:

```
ma_dziecko(X1) :- rodzic(X1, _).  
dziecko(X2, Y) :- rodzic(Y, X2).
```

Reprezentacja danych w Prologu

- Dane złożone (strukturalne) - dane, które składają się z kilku elementów, tworzących jeden obiekt; w szczególności składowymi *struktury* mogą być również dane złożone
- Dane złożone tworzone są za pomocą *funktora* i *argumentów* traktowanych razem jako jeden obiekt programu

Przykład

Zapis daty w Prologu

```
date(1, october, 2000)
```

Reprezentacja danych w Prologu

Składowymi (argumentami) danych złożonych (strukturalnych) mogą być zarówno *stale*, jak i *zmienne* prologowe:

Przykład

Dowolny dzień marca 2000 roku:

`date (Day , march , 2000)`

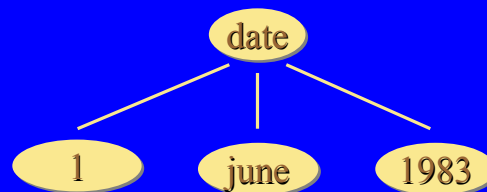


Reprezentacja danych w Prologu

Wszystkie dane strukturalne w Prologu mogą być reprezentowane za pomocą *drzew*, którego korzeniem jest funktor a węzłami potomnymi - jego argumenty

Przykład

`date (1 , june , 1983)`



Reprezentacja danych w Prologu

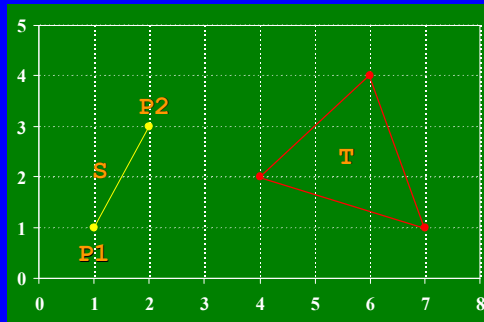
Przykład reprezentacji danych strukturalnych

P1 jako `point(1,1)`

P2 jako `point(2,3)`

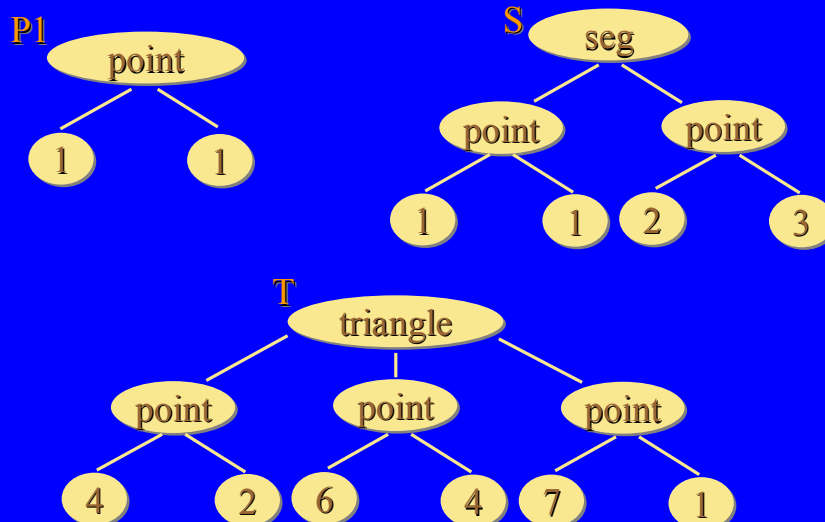
S jako `seg(P1,P2)` czyli `seg(point(1,1),point(2,3))`

T jako `triangle(point(4,2),point(6,4),point(7,1))`



Reprezentacja danych w Prologu

Drzewiasta reprezentacja tych danych



Reprezentacja danych w Prologu

Dane strukturalne określone są zarówno przez nazwę funktora, jak i liczbę argumentów tzw. *arność*. Oznacza to, iż dwa funktory o tej samej nazwie, lecz innej arności są różnymi obiektami.

Przykład

Punkt w przestrzeni 2-wymiarowej:

point(7,3)

funktor i
2 argumenty

punkt w przestrzeni 3-wymiarowej:

point(7,3,1)

funktor i
3 argumenty

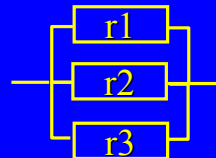
Reprezentacja danych w Prologu

Inny przykład zastosowania struktur

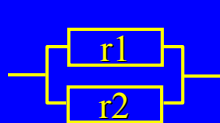
seq(r1, r2)



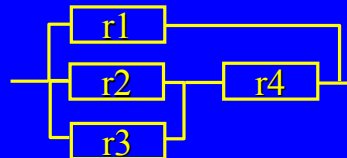
par(r1, par(r2, r3))



par(r1, r2)



par(r1, seq(par(r2, r3), r4))



Reprezentacja danych w Prologu

Pojęcie formalne termu

Do termów zaliczamy:

- stałe i zmienne prologowe
- struktury reprezentowane za pomocą funktorów o dowolnej arności, których argumentami są inne termy

Mechanizm uzgadniania (ang. *unification*)

O uzgodnieniu dwóch termów mówimy wtedy, gdy:

- termy są *identyczne* lub,
- zmienne zawarte w obu termach mogą przyjąć wartości, dla których termy te staną się identyczne

Przykład

Termy: **date (D, M, 1993)** i **date (D1, may, Y1)**
można uzgodnić, gdy:

D przyjmie wartość **D1**

M przyjmie wartość **may**

Y1 przyjmie wartość **1993**

Mechanizm uzgadniania (ang. *unification*)

Proces uzgadniania może zakończyć się *porażką*, kiedy nie istnieje żadne możliwe podstawienie zmiennych, które spowodowałoby, iż termy będą identyczne.

Przykład

Termów: `date(D,M,1993)` i `date(D1,M1,1644)` nie można dopasować, bo: `1993` i `1644`, to różne stałe numeryczne.

Podobnie `date(X,Y,Z)` i `point(X,Y,Z)`, bo: `date` i `point`, to różne funktory.

Mechanizm uzgadniania (ang. *unification*)

Operator uzgadniania w Prologu oznaczany jest znakiem '='. Proces uzgadniania może być *jawnym* elementem zapytań i kodu programu prologowego.

Przykład

?- `date(D,M,1993)=date(D1,may,Y1)` .
można uzgodnić, gdy: `D=1, D1=1, M=may, Y1=1993`
albo `D=third, D1=third, M=may, Y1=1993` albo

...
lecz podstawienia te nie są *najogólniejsze*, w przeciwieństwie do:
`D=D1, M=may, Y1=1993`

Mechanizm uzgadniania (ang. *unification*)

Mechanizm uzgadniania w Prologu zawsze wyznacza *najogólniejszy możliwy unifikator*, tzn. taki zbiór podstawień zmiennych, który w najmniejszym możliwym stopniu ogranicza zbiór wartości zmiennych.

Przykład

?- `date(D,M,1993)=date(D1,may,Y1),`
`date(D,M,1993)=date(15,M,Y).`

najpierw:	D=D1	ostateczny	D=15
	M=may	rezultat:	D1=15
	Y1=1993		M=may
			Y1=1993
			Y=1993

Mechanizm uzgadniania (ang. *unification*)

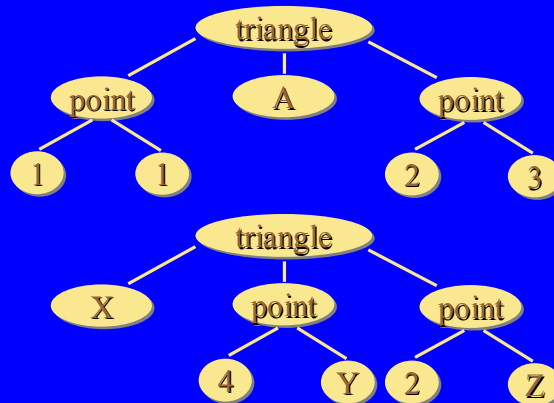
Przebieg procesu uzgadniania termów S i T w Prologu:

- Jeżeli dwa termy S i T, są stałymi, to uzgodnienie zachodzi, gdy są identyczne,
- Jeżeli S jest zmienną a T dowolnym termem, to uzgodnienie zachodzi, gdy S przypiszemy wartość T; podobnie w sytuacji odwrotnej, kiedy T jest zmienną, a S dowolnym termem, to uzgodnienie zajdzie, gdy T przypiszemy wartość S
- Jeżeli S i T są obiektami złożonymi, to uzgodnienie zachodzi, gdy:
 - S i T mają ten sam funktor, i
 - zachodzi uzgodnienie pomiędzy wszystkimi ich składowymi

Ostateczny rezultat zależy wtedy od uzgodnienia poszczególnych składowych - jest złożeniem ich uzgodnień .

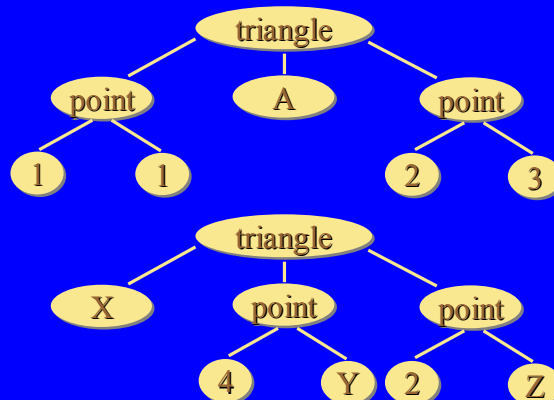
Mechanizm uzgadniania (ang. *unification*)

?- `triangle(point(1,1),A,point(2,3)) = triangle(X,point(4,Y),point(2,Z))`.



Mechanizm uzgadniania (ang. *unification*)

Proces uzgadniania przebiega *rekurencyjnie* od korzenia do liści struktur drzewiastych, reprezentujących dopasowywane terminy.



Mechanizm uzgadniania (ang. *unification*)

Proces uzgadniania przebiega od korzenia do liści struktur drzewiastych, reprezentujących dopasowywane terminy.

Poszczególne etapy procesu uzgadniania:

```
triangle=triangle,  
point(1,1)=X,  
A=point(4,Y),  
point(2,3)=point(2,Z).
```

Sukces uzgodnienia końcowego wynika z udanych uzgodnień poszczególnych składowych terminów:

```
X=point(1,1)  
A=point(4,Y)  
Z=3
```

Mechanizm uzgadniania (ang. *unification*)

Proces uzgadniania może być wykorzystany sam w sobie do przeprowadzenia pewnych „obliczeń”.

Przykład

Sprawdzenie czy odcinek jest pionowy, czy też poziomy?

Zapis własności za pomocą klauzul w Prologu:

```
vertical(seg(point(X,Y1),point(X,Y2))).  
horizontal(seg(point(X1,Y),point(X2,Y))).
```

Zapytanie w Prologu:

```
?- vertical(seg(point(1,1),point(1,2))).  
Yes  
?- horizontal(seg(point(1,1),point(2,Y))).  
Y=1
```

Mechanizm uzgadniania (ang. *unification*)

Zapytanie bardziej ogólne:

Czy istnieje odcinek pionowy zaczynający się w punkcie (2,3)?

?- **vertical**(**seg**(**point**(2,3),P)).

P=point(2,Y)

Zapytanie jeszcze ogólniejsze:

Czy istnieje odcinek zarówno pionowy, jak i poziomy?

?- **vertical**(S),**horizontal**(S).

S=seg(**point**(X,Y),**point**(X,Y))

odpowiedź jest twierdząca, gdyż każdy odcinek „zdegenerowany” do punktu jest zarówno pionowy, jak i poziomy.

Związki języka Prolog z logiką formalną

Podstawowe pojęcia

- logika predykatów I rzędu
- klauzule Horn’a
- skolemizacja
- zasada rezolucji
- pojęcie najogólniejszego unifikatora (MGU)
- mechanizm unifikacji (uzgadniania)

Podsumowanie

- Proste obiekty w Prologu to *atomy*, *stałe*, *zmiennie*. Obiekty proste i złożone (strukturalne) to *termy*.
- Termy składają się z *funktora* (*operatora*, *symbolu funkcyjnego*), określonego przez nazwę i arność oraz jego argumentów
- Typ obiektu jest rozpoznawany jedynie w oparciu o jego budowę składniową (brak mechanizmu deklaracji)
- Zakresem leksykalnym zmiennej jest jedna klauzula, zatem zmienne o tej samej nazwie w różnych klauzulach są różne
- Termy można reprezentować w postaci drzew, zaś ich przetwarzanie traktować jako przetwarzanie struktur drzewiastych

Podsumowanie

- Proces uzgadniania polega na sprawdzeniu czy dwa termy są identyczne lub stwierdzeniu dla jakich podstawień zmiennych termy będą takie same
- Jeżeli proces uzgadniania zmiennych zakończy się sukcesem, to w wyniku otrzymujemy *najogólniejsze* możliwe podstawienia zmiennych