Wprowadzenie do Sztucznej Inteligencji
Laboratorium lista 0.1
Elementy języka PROLOG: fakty i zapytania

Przemysław Kobylański

Część I
Wprowadzenie

1 Stałe i zmienne

Jedynym dostępnym w języku PROLOG rodzajem danych są termy, które dzielimy na proste i złożone.

Termami prostymi są stałe i zmienne.

Najczęściej stałą zapisywać będziemy jako identyfikator alfanumeryczny rozpoczynający się małą literą ale może być nim liczba (całkowita lub zmiennopozycyjna) albo dowolny ciąg znaków ujęty w apostrofy.

Przykłady stałych:

- `a`, `clean`, `bogactwo`, `u2`
- 10, 123
- 3.14, $1e-15$
- `'Jan Kowalski'`, `'Konstytucja 3 maja'`

Stałe używać będziemy do reprezentowania obiektów. Na przykład jeśli chcemy wyrazić w programie duże czerwone pudełko, to możemy zapisać je w postaci stałej `bigRedBox` albo `redBox` albo `box1`. Ważne aby dany obiekt był w całym programie oznaczony jedną stałą i żaden inny obiekt nie był wyrażony tą stałą.

Zmienne zapisuje się w postaci identyfikatorów alfanumerycznych rozpoczynających się wielką literą. Szczególną postać mają tzw. zmienne anonimowe, których nazwy rozpoczynają się od znaku podkreślenia.

Przykłady zmiennych:

- `X`, `NowyStan`
- `_`
- `_to_jest_nieistotne`

Zmienna w PROLOGU charakteryzuje się tym, że nie jest zmienna tj. nie może zmienić raz przyjętej wartości. Taka własność „niezmiennych zmiennych" jest charakterystyczna nie tylko dla PROLOGU ale spotyka się w wielu językach programowania funkcyjnego (np. ERLANG).

```
rodzice(uranus, gaia, rhea).
rodzice(uranus, gaia, cronus).
rodzice(cronus, rhea, zeus).
rodzice(cronus, rhea, hera).
rodzice(cronus, rhea, demeter).
rodzice(zeus, letom, artemis).
rodzice(zeus, leto, apollo).
rodzice(zeus, demeter, persephone).
```

Rysunek 1: Garść faktów o pewnej rodzinie.

2 Predykaty i fakty

Program w Prologu wyraża logiczne warunki czyli predykaty.

Niech p będzie nazwą n-argumentowego predykatu (warunku) a $c_1, c_2, \ldots, c_n$ będą stałymi reprezentującymi n obiektów. Wówczas formuła $p(c_1, c_2, \ldots, c_n)$ stwierdza, że obiekty reprezentowane stałymi $c_1, c_2, \ldots c_n$ spełniają warunek p.

Pisząc p/n informujemy, że predykat (warunek) p jest n-argumentowy.

Najprostszym sposobem wyrażenia predykatu jest wymienienie faktów w postaci formuł atomowych, należy przy tym pamiętać, że każdy fakt powinien być zakończony kropką.

Koniecznie przypomnij sobie pojęcie formuł atomowych z rachunku predykatów, którego podstawy poznałeś na kursie **Logika i struktury formalne**.

Na rysunku 1 przedstawiono przykład faktów opisujących predykat `rodzice/3`:

$$rodzice(Ojciec, Matka, Dziecko)$$

jaki zachodzi w pewnej rodzinie.

3 Zapytania

Załóżmy, że w pliku `olimp.pl` znajdują się fakty w takiej postaci jak przedstawiono na rysunku 1.

Po uruchomieniu programu SWI-Prolog należy wgrać (dokładniej skompilować) fakty składające się na program `olimp.pl` wykonując polecenie:

```
?- consult('olimp.pl').
```

albo krócej:

```
?- [olimp].
```

Jeśli kompilacja była bezbłędna możemy przystąpić do zadawania systemowi Prolog pytań dotyczących wiedzy zapisanej w postaci skompilowanych formuł.

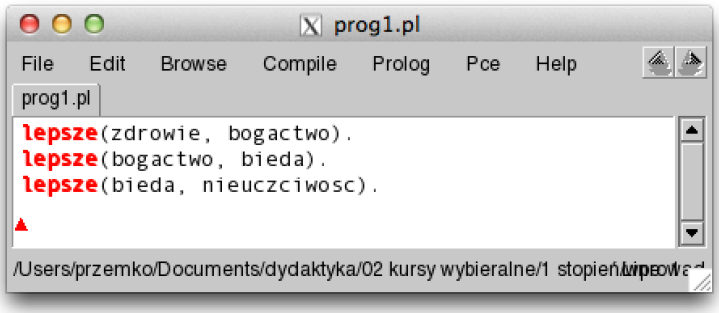
Na początek zapytajmy się czy Uran jest ojcem Zeusa:

```
?- rodzice(uranus, _, zeus).
false.
```

Jak widać otrzymaliśmy odpowiedź negatywną. Zwróć uwagę na użycie anonimowej zmiennej `_` dla nieistotnego dla nas imienia matki Zeusa.

Zadajmy zatem pytanie który z bogów `X` jest ojcem Zeusa:

```
?- rodzice(X, _, zeus).
X = cronus.
```



Rysunek 2: Okno edytora Emacs z wpisanymi trzema faktami.

Warunki w pytaniu można łączyć spójnikiem koniunkcji (przecinek) albo alternatywy (średnik).

W następujący sposób możemy zapytać się o to, która z bogiń `X` jest babką Apolla ze strony ojca:

```
?- rodzice(Y, _, apollo), rodzice(_, X, Y).
Y = zeus,
X = rhea.
```

Przy okazji dowiedzieliśmy się, że ojcem Apolla jest Zeus.

Zadajmy teraz pytanie jakie dzieci ma Zeus:

```
?- rodzice(zeus, _, X).
X = artemis ;
X = apollo ;
X = persephone.
```

Po każdej odpowiedzi naciśnij średnik aby poznać kolejną odpowiedź. Zauważ, że po uzyskaniu ostatniej odpowiedzi `X = persephone` system wydrukował prompt `?-` i czeka na kolejne pytanie.

4 Edycja i kompilacja programu

Edytor Emacs uruchamia się w systemie PROLOG predykatem `emacs/1`. Jego argumentem wywołania jest nazwa pliku (pamiętaj o rozszerzeniu `.pl` i apostrofach).

Gdy wydamy polecenie:

```
?- emacs('prog1.pl').
```

pojawia się okno edytora. Wpiszmy w nim następujące trzy fakty:

```
lepsze(zdrowie, bogactwo).
lepsze(bogactwo, bieda).
lepsze(bieda, nieuczciwosc).
```

Na rysunku 2 przedstawiono widok okna z wpisanymi powyższymi faktami.

Aby skompilować program należy wybrać z menu `Compile` opcję `Compile buffer` albo nacisnąć kombinację `Control-c` a potem kombinację `Control-b`.

Po skompilowaniu programu `prog1.pl` można zadać pytanie jakie `X` jest lepsze od nieuczciwości:

```
?- lepsze(X, nieuczciwosc); lepsze(X, Y), lepsze(Y, nieuczciwosc);
   lepsze(X, Y), lepsze(Y, Z), lepsze(Z, nieuczciwosc).
X = bieda ;
X = bogactwo,
Y = bieda ;
X = zdrowie,
Y = bogactwo,
Z = bieda ;
false.
```

Nie zapomnij po każdej odpowiedzi nacisnąć średnik. Po ostatniej odpowiedzi pojawi się informacja `false` wskazująca na to, że nie ma już więcej odpowiedzi.

5 Dodawanie i usuwanie faktów

Aby podczas działania programu mieć możliwość dodawania lub usuwania faktów definiujących predykat p/n musimy zadeklarować w pliku źródłowym, że jest on dynamiczny. W tym celu używamy dyrektywy `dynamic`.

Załóżmy, że w pliku `wiedza.pl` zapisano fakty wyrażające stan wiedzy studenta po pierwszym semestrze studiów:

```
:- dynamic znam/1.

znam(algebra).
znam(analiza).
znam(logika).
```

Jak widać predykat `znam/1` został zadeklarowany jako dynamiczny. Dzięki temu podczas działania programu możemy modelować zarówno poszerzanie stanu wiedzy (dodawanie nowych faktów) jak i zapominanie części wiedzy (usuwanie faktów).

Załóżmy, że skompilowano program `wiedza.pl`.

Zadając następujące pytanie poznamy co umie student po pierwszym semestrze:

```
?- znam(X).
X = algebra ;
X = analiza ;
X = logika.
```

Załóżmy teraz, że skończył się drugi semestr studiów i student nauczył się języka Java ale zapomniał czego uczył się na kursie analizy matematycznej. Należy zatem dokonać zmian w już skompilowanym programie.

Po pierwsze dodamy nowy fakt, że student zna jęzk Java. Użyjemy w tym celu predykat `assert/1`:

```
?- assert(znam(java)).
```

Po drugie usuniemy z bazy faktów informację o znajomości analizy matematycznej. Użyjemy w tym celu predykat `retract/1`:

```
?- retract(znam(analiza)).
```

O aktualnym stanie wiedzy studenta możemy przekonać się zadając następujące pytanie:

```
?- znam(X).
X = algebra ;
X = logika ;
X = java.
```

Jeśli chcemy dopisać nowy fakt na początku bazy faktów definiujących dany predykat, to należy użyć predykat `asserta/1`, natomiast jeśli na końcu, to predykat `assertz/1`. Predykat `assert/1` dopisuje nowy fakt zawsze na końcu bazy faktów i zaleca się stosowanie w jego miejsce predykat `assertz/1`.

Możliwe jest również usunięcie wszystkich faktów pasujących do zadanego wzorca stosując predykat `retractall/1`.

Dla przykładu gdybyśmy chcieli usunąć wszystkie fakty $p/2$, w których drugim argumentem jest stała c, to użylibyśmy `retractall(p(_, c))`.

Możliwość dodawania i usuwania warunków z już skompilowanego programu podczas jego działania umożliwia tworzenie samomodyfikujących się programów. Daje to potężną moc ale często obracającą się przeciw programiście, więc należy stosować ją z umiarem[1].

W naszych programach będziemy ograniczać się do dodawania i usuwania jedynie faktów stanowiących bazę wiedzy o aktualnym stanie modelowanego systemu (stanu agenta, stanu środowiska itp.).

Program w Prologu wnioskując logicznie zdobywa wiedzę, usuwa z siebie tę już zdezaktualizowaną i dopisuje nowo nabytą. Czy to już jest **sztuczna inteligencja**?

Część II
Zadania i polecenia

Polecenie 1

Opis użytych predykatów

Predykat `true` wyraża prosty warunek, który jest zawsze spełniony.

Zadaj pytanie

```
?- true.
```

Wyjaśnienie

System odpowiada **true** gdyż warunek `true` jest spełniony.

Polecenie 2

Opis użytych predykatów

Predykat `fail` zawsze zawodzi.

Zadaj pytanie

```
?- fail.
```

Wyjaśnienie

System odpowiada `false` ponieważ warunek `fail` zawiódł.

[1] Kto mieczem wojuje ten od miecza ginie.

Polecenie 3

Opis użytych predykatów

Do wyrażenia koniunkcji warunków stosuje się przecinek.

Zadaj pytanie

```
?- true, fail.
```

Wyjaśnienie

System odpowiada `false` ponieważ nie jest prawdą, że oba warunki `true` i `fail` zachodzą (koniunkcja jest prawdziwa tylko w przypadku kiedy jej oba człony są prawdziwe).

Polecenie 4

Opis użytych predykatów

Do wyrażenia alternatywy warunków stosuje się średnik.

Zadaj pytanie

```
?- true; fail.
```

Wyjaśnienie

System odpowiada **true** ponieważ przynajmniej pierwszy człon alternatywy jest spełniony. Po słowie **true** system czeka na decyzję użytkownika. Jeśli naciśniemy klawisz **[Enter]**, to pojawi się kolejny prompt jako zachęta do wprowadzenia kolejnego pytania.

Jeśli natomiast naciśniemy średnik, to system przystąpi do poszukiwania innego alternatywnego rozwiązania ale że drugi człon alternatywy nie jest spełniony (warunek `fail`), to pojawia się odpowiedź `false`.

Polecenie 5

Opis użytych predykatów

Warunki można grupować korzystając z nawiasów.

Zadaj pytanie

```
?- (true, fail); (fail; true).
```

Wyjaśnienie

System odpowiada **true** ponieważ chociaż pierwszy człon alternatywy (koniunkcja `true, fail`) jest fałszywy, to jednak drugi jej człon (alternatywa `fail; true`) jest prawdziwy.

Polecenie 6

Opis użytych predykatów

W Prologu można korzystać ze zmiennych, przy czym ich nazwy pisze się z wielkiej litery.

Predykat `=` wyraża równość.

Zadaj pytanie

```
?- X = 1.
```

Wyjaśnienie

System odpowiada `X = 1` ponieważ liczba 1 jest jedyną wartością spełniającą warunek `X = 1`.

Polecenie 7

Zadaj pytanie

```
?- 1 = X.
```

Wyjaśnienie

Tym razem również system odpowiedział `X = 1`. Zwróć uwagę, że predykat `=` nie wyraża podstawienia ale zunifikowanie.

Polecenie 8

Zadaj pytanie

```
?- X = 1, X = 2.
```

Wyjaśnienie

System odpowiada `false` ponieważ żadna wartość nie jest jednocześnie równa dwóm różnym liczbom 1 i 2.

Polecenie 9

Zadaj pytanie

```
?- X = 1; X = 2.
```

Wyjaśnienie

System drukuje odpowiedź `X = 1` i czeka na to co zrobi użytkownik. Jeśli naciśniesz klawisz **[Enter]**, to pojawi się zachęta do wpisania następnego pytania.

Jednak wartość 1 nie jest jedyną spełniającą alternatywę dwóch warunków $X = 1 \vee X = 2$. Jeśli naciśniesz średnik, to system poszuka drugiej odpowiedzi i pojawi się `X = 2`.

Polecenie 10

Opis użytych predykatów

Do obliczania wartości wyrażenia arytmetycznego służy predykat `is`. W wyrażeniu arytmetycznym mogą wystąpić operacje `+, -, *, /, **` (podnoszenie do potęgi), stałe liczbowe oraz zmienne, którym nadano już wcześniej wartości będące liczbami.

Zadaj pytanie

```
?- X is 1, Y is X+1.
```

Wyjaśnienie

System udziela odpowiedzi `X = 1, Y = 2`. Najpierw pod zmienną `X` została podstawiona wartość 1 (zmienna `X` została zunifikowana z wartością 1), a następnie obliczona wartość wyrażenia `1+1` została zunifikowana ze zmienną `Y`.

Uwaga

W języku Prolog nie jest możliwa zmiana wartości zmiennej. Dla przykładu w języku Pascal po wykonaniu instrukcji `x := 1; x := x+1` zmienna `x` będzie równa 2, natomiast w Prologu po zadaniu pytania `X is 1, X is X+1` otrzymamy odpowiedź `false`, gdyż żadna wartość `X` nie może być większa o 1 od samej siebie (zmienna `X` ma już wartość 1 a nie da się zunifikować ze sobą dwóch różnych wartości 1 i 2).

Polecenie 11

Opis użytych predykatów

Wartości wyrażeń arytmetycznych można porównywać za pomocą warunków `=:=` (równe), `=\=` (różne), `<, =<, >, >=`.

Zadaj pytanie

```
?- (X is 1; X is 2), 2*X > 3.
```

Wyjaśnienie

System poszukuje takiej wartości `X`, która podwojona jest większa od 3. Do wyboru ma dwie możliwe wartości 1 lub 2 (alternatywa `X is 1; X is 2`). Pierwsza z wartości nie spełnia zadanego warunku, natomiast druga go spełnia. Stąd odpowiedź `X=2`.

Polecenie 12

Opis użytych predykatów

Predykat `between(A, B, X)` jest spełniony gdy wartość `X` jest liczbą całkowitą z zakresu od `A` do `B`. Przy wywołaniu predykatu `between` wartości `A` i `B` muszą być liczbami całkowitymi natomiast `X` może być liczbą całkowitą lub zmienną.

Zadaj pytanie

```
?- between(7, 11, K), K =\= 9.
```

Wyjaśnienie

System udziela czterech odpowiedzi, gdyż w zakresie od 7 do 11 są cztery liczby całkowite różne od 9.

Zadanie 13

Zadaj systemowi Prolog pytanie, na które odpowiedzi są rozwiązaniem następującego zadania:

> *Jakie trójkąty prostokątne można skonstruować wybierając trzy jego boki spośród sześciu odcinków o długościach będących liczbami całkowitymi z zakresu od 1 do 6.*

Zadanie 14

Jeszcze raz rozwiąż poprzednie zadanie ale dla większej liczby odcinków (np. 20 o długościach od 1 do 20). Postaraj się wyeliminować rozwiązania równoważne (np. rozwiązania $3, 4, 5$ i $4, 3, 5$ są równoważne).

Polecenie 15

Umieść w pliku `prog2.pl` dyrektywę `dynamic` i następujące fakty:

```
:- dynamic p/1.

p(a).
p(b).
p(c).
```

Polecenie 16

1. Skompiluj program z pliku `prog2.pl` i zadaj pytanie tak aby poznać wszystkie wartości `X` spełniające warunek `p(X)`.
2. Usuń, stosując predykat `retract/1`, fakt `p(b)`.
3. Ponownie zadaj pytanie aby poznać jakie tym razem wartości `X` spełniają warunek `p(X)`.
4. Dodaj, stosując predykat `assert/1`, fakt `p(b)`.
5. I jeszcze raz zadaj pytanie aby poznać jakie tym razem wartości `X` spełniają warunek `p(X)` (zwróć uwagę na kolejność uzyskiwanych odpowiedzi).

Polecenie 17

Opis użytych predykatów

Predykat `halt` służy do zakończenia pracy z Prologiem.

Zadaj pytanie

```
?- halt.
```