Implementacja i analiza predykatów oraz prostego systemu ekspertowego w Prologu

Mateusz Łopaciński

10 maja 2025

1 Zadanie 1: Predykaty na listach

1.1 Treść zadania

Zdefiniować w języku Prolog dwa predykaty:

1. **elem/2** — sprawdzający, czy element należy do listy.
2. **dlugosc/2** — obliczający długość listy.

1.2 Rozwiązanie

```
% elem(X, L)

elem(X, [X|_]).
elem(X, [_|Tail]) :-
    elem(X, Tail).

% dlugosc(L, N)

dlugosc([], 0).
dlugosc([_|Tail], N) :-
    dlugosc(Tail, M),
    N is M + 1.
```

Listing 1: Definicje predykatów elem/2 oraz dlugosc/2

1.3 Opis działania predykatów

- **elem/2**:
  Predykat `elem(X, L)` sprawdza, czy element `X` występuje w liście `L`.
  - Przypadek podstawowy: Element znajduje się na pierwszej pozycji (głowie listy).
  - Przypadek rekurencyjny: Element jest sprawdzany w dalszej części listy (ogonie).

- **dlugosc/2**:
  Predykat `dlugosc(L, N)` oblicza długość listy `L`.
  - Przypadek podstawowy: Pusta lista (`[]`) ma długość `0`.
  - Przypadek rekurencyjny: Dla każdego elementu dodajemy `1` do długości ogona listy.

1.4 Przykłady użycia

- Sprawdzanie obecności elementu:
  - `?- elem(3, [1,2,3,4]).` — **true**.
  - `?- elem(5, [1,2,3,4]).` — **false**.
- Obliczanie długości listy:
  - `?- dlugosc([a,b,c,d], L).` — $\mathbf{L = 4}$.
  - `?- dlugosc([], L).` — $\mathbf{L = 0}$.

2 Zadanie 2: Prosty system ekspertowy

2.1 Treść zadania

Zaimplementować prosty system ekspertowy, który określa, czy dana sieć jest bezpieczna. Wnioskowanie powinno opierać się na co najmniej 5 regułach i 10 faktach.

2.2 Rozwiązanie

```
network_status(up).
ping_sweep_detected(no).
port_scan_detected(no).
ddos_attack_detected(no).
firewall_enabled(yes).
antivirus_updated(yes).
intrusion_detection_active(yes).
recent_security_audit(passed).
open_ports(few).
admin_password(strong).

network_secure :-
    network_status(up),
    ping_sweep_detected(no),
    port_scan_detected(no),
    ddos_attack_detected(no),
    firewall_enabled(yes),
    antivirus_updated(yes),
    intrusion_detection_active(yes).

network_vulnerable :-
```

```
    ping_sweep_detected(yes);
    port_scan_detected(yes);
    ddos_attack_detected(yes).

needs_attention :-
    firewall_enabled(no);
    antivirus_updated(no);
    intrusion_detection_active(no).

audit_recommended :-
    recent_security_audit(failed);
    open_ports(many).

strong_admin_security :- admin_password(strong).
weak_admin_security :- admin_password(weak).
```

Listing 2: Prosty system ekspertowy (statyczna baza wiedzy)

2.3 Przykłady użycia

- Sprawdzanie bezpieczeństwa sieci:
  - `?- network_secure.` — **true**.
- Sprawdzanie podatności sieci:
  - `?- network_vulnerable.` — **false**.
- Sprawdzanie konieczności audytu:
  - `?- audit_recommended.` — **false**.