

Prosta baza wiedzy i wnioskowanie w Prologu

Karolina Łukasik

9 czerwca 2023

Spis treści

1	Prosta sieć semantyczna	1
2	Przykłady wnioskowania i unifikacji	2
3	Rozwiązywanie problemów	4
4	Przykłady działania reguł diagnostycznych	5
5	Dodatek: uruchamianie ekspresu	6

1 Prosta sieć semantyczna

W poniższym raporcie skonstruujemy prostą bazę wiedzy obsługi ekspresu do kawy. Pierwszym krokiem będzie stworzenie prostej sieci semantycznej, w której zdefiniujemy, czym jest ekspres, z czego się składa i jakie czynności można na nim wykonywać. Zdefiniujemy również trzy postacie, które będą miały różne uprawnienia w posługiwaniu się urządzeniem.

```
% SEMANTIC NETWORK

device(coffee_machine).

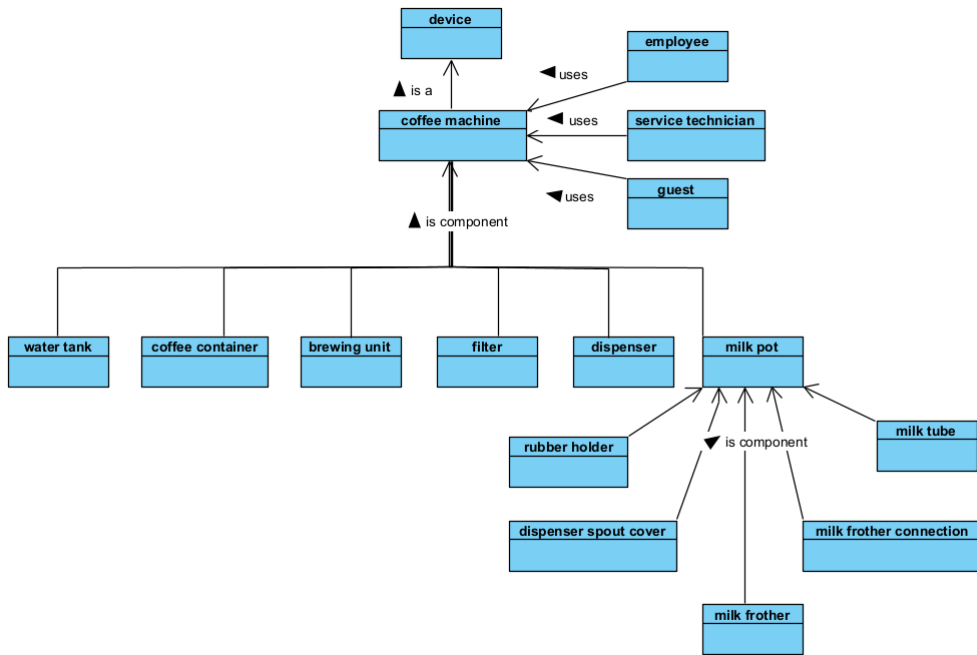
item(water_tank).
item(coffee_container).
item(brewing_unit).
item(filter).
item(dispenser).
item(milk_pot).

item_component(milk_tube).
item_component(rubber_holder).
item_component(milk_frother).
item_component(milk_frother_connection).
item_component(dispenser_spout_cover).

user(service_technician).
user(employee).
user(guest).
```

Rysunek 1: Prosta sieć semantyczna w Prologu

Jak widać na powyższym diagramie, sieć semantyczną możemy przedstawić w formie etykietowanego digrafu. Poniżej natomiast możemy się przyjrzeć zależnościom zdefiniowanym za pomocą termów złożonych.



Rysunek 2: Diagram sieci semantycznej

```

consists_of(coffee_machine, water_tank).
consists_of(coffee_machine, coffee_container).
consists_of(coffee_machine, brewing_unit).
consists_of(coffee_machine, filter).
consists_of(coffee_machine, dispenser).
consists_of(coffee_machine, milk_pot).

consists_of(milk_pot, milk_tube).
consists_of(milk_pot, rubber_holder).
consists_of(milk_pot, milk_frother).
consists_of(milk_pot, milk_frother_connector).
consists_of(milk_pot, dispenser_spout_cover).

services(coffee_machine, service_technician).
makes_coffee(coffee_machine, employee).

```

Rysunek 3: Termy złożone opisujące relacje

2 Przykłady wnioskowania i unifikacji

Pokażemy teraz na kilku prostych przykładach, w jaki sposób można wykorzystać mechanizm unifikacji i dopasowania na bazie wiedzy skonstruowanej w poprzedniej sekcji. Na rysunku 4 zawartych jest kilka klauzul, z których będziemy korzystać.

```

is_component(X,Y) :- consists_of(Y,X).
is_component(X,Y) :-
    consists_of(Z, X),
    is_component(Z,Y).

is_component(X) :- consists_of(A, X), \+ consists_of(X, A).
is_device(X) :- consists_of(X, _), \+ consists_of(_, X).
uses(X, Y) :- services(Y, X); makes_coffee(Y,X).

```

Rysunek 4: Przykładowe klauzule

Za ich pomocą możemy wyciągnąć interesujące informacje np. poprzez podstawienie do zmiennej lub sprawdzenie, czy dany term jest zgodny z prawdą. Ciekawym zabiegiem, który można wykorzystać podczas definiowania klauzul jest rekurencyjne sprawdzanie zależności. Powyżej ma to miejsce w klauzuli *is_component*, która jest prawdziwa, jeśli element jest bezpośrednim podkomponentem danej części lub podkomponentem podkomponentu na dowolnej głębokości. Poza tym w ostatniej klauzuli *uses* wykorzystujemy alternatywę, a w przedostatniej *is_device* zaprzeczenie. Klauzula sprawdza, czy warunki po prawej stronie zostają spełnione i próbuje dopasować wartości do zmiennych lub zwraca wartość true lub false. Poniżej przykłady zwróconych wartości:

```

is_component(X, coffee_machine) = is_component(water_tank,
coffee_machine).
X = water_tank

consists_of(milk_pot, Y) = consists_of(milk_pot, dispenser_spout_cover).
Y = dispenser_spout_cover

```

Rysunek 5: Przykłady podstawienia pod zmienną

```

is_device(filter).
false

is_component(filter).
true

consists_of(_, milk_tube)
true

consists_of(X, milk_tube)
X = milk_pot

consists_of(milk_tube, _)
false

```

Rysunek 6: Inne przykłady unifikacji w tym sprawdzanie prawdziwości

```

is_component(X, coffee_machine).
X = water_tank
X = coffee_container
X = brewing_unit
X = filter
X = dispenser
X = milk_pot
X = milk_tube
X = milk_frother
X = milk_frother_connector
X = dispenser_spout_cover

```

Rysunek 7: Przykład na wielokrotne podstawienie

3 Rozwiązywanie problemów

Kolejnym krokiem będzie zdefiniowanie reguł diagnostycznych i możliwych rozwiązań problemów, które mogą się pojawić podczas korzystania z ekspresu. Zapiszemy je w formie klauzul oraz predykatów prologowych. Zaimplementujemy poniższą listę problemów oraz ich rozwiązań:

Tabela 1: Możliwe problemy i ich rozwiązania

Problem	Rozwiązanie
Nie można wyjąć jednostki zaparzającej	Zamknij klapkę serwisową. Wyłącz urządzenie i włącz je ponownie. Poczekaaj, aż na wyświetlaczu pojawi się komunikat „Machine ready” (Urządzenie jest gotowe), i wtedy wyjmij jednostkę zaparzającą.
Nie można włożyć jednostki zaparzającej	Przed włożeniem jednostka zaparzająca nie była w prawidłowej pozycji. Upewnij się, że dźwignia przylega do podstawy jednostki zaparzającej i że zaczep jednostki zaparzającej znajduje się w prawidłowej pozycji.
Rozwodniona kawa	Zmień ustawienie na drobniejsze mielenie.
Zbyt zimne	Wyczyść wylewkę dozownika kawy i jej otwory za pomocą wycioru.
Kawa nie wypływa	Zamknij klapkę serwisową. Wyłącz urządzenie i włącz je ponownie. Poczekaaj, aż na wyświetlaczu pojawi się komunikat „Machine ready” (Urządzenie jest gotowe), i wtedy wyjmij jednostkę zaparzającą.
Kawa wypływa wolno	Użyj innej mieszanki kawy lub wyreguluj młynek.
Zbyt zimne mleko	Spienione mleko jest za zimne.
Mleko nie jest spieniane	Zamknij klapkę serwisową. Wyłącz urządzenie i włącz je ponownie. Poczekaaj, aż na wyświetlaczu pojawi się komunikat „Machine ready” (Urządzenie jest gotowe), i wtedy wyjmij jednostkę zaparzającą.
Filtr nie pasuje	Wypuść pęcherzyki powietrza z filtra.

Powyższe problemy zaimplementujemy w formie termów złożonych, do których Prolog będzie próbował się dopasować szukając rozwiązań. Implementacja w każdym z przypadków będzie analogiczna. Poniżej przykład dwóch z nich:

```

solution(
    problem(coffee_flows_slow),
    applies_to(dispenser),
    'Use a different coffee blend or adjust the grinder.'
).

solution(
    problem(too_cold_milk),
    applies_to(milk_frother_connector),
    'The frothed milk is too cold.'
).

```

Rysunek 8: Implementacja reguł diagnostycznych

4 Przykłady działania reguł diagnostycznych

Na kilku poniższych przykładach możemy zaobserwować możliwości wykorzystania opracowanej bazy wiedzy przy rozwiązywaniu problemów. Wystarczy, iż do predykatu *problem_cause* podstawimy dany problem, a Prolog za pomocą zdefiniowanej bazy wiedzy wywnioskuje resztę

potrzebnych informacji. Dodatkowo pomocniczy predykat *print_sol* zaimplementowany został tak by w czytelny sposób drukować informacje dotyczące problemu. Wypisuje on kolejno:

- jakiego podkomponentu dotyczy problem,
- możliwe rozwiązanie problemu,
- czy element jest podkomponentem jakiegoś większego elementu i jeśli tak, to wypisuje rekurencyjnie wszystkie podkomponenty aż do głównej maszyny.

```

⚙️ problem_cause(too_cold_milk).
The problem applies to:
Item: milk_frother_connector
Probable solution to the problem:
The frothed milk is too cold.
Element is also a component:
milk_pot
coffee_machine
false

```

Rysunek 9: Przykłady dopasowania w regułach diagnostycznych

```

⚙️ problem_cause(brew_unit_cannot_be_inserted).
The problem applies to:
Item: brewing_unit
Probable solution to the problem:
Before inserting the brewing unit
  was not in the correct position. Make sure
  that the lever rests against the base
  the brew unit and that the catch
  the brewing unit is in
  correct position.
Element is also a component:
coffee_machine
false

⚙️ problem_cause(coffee_flows_slow).
The problem applies to:
Item: dispenser
Probable solution to the problem:
Use a different coffee blend or adjust the grinder.
Element is also a component:
coffee_machine
false

```

Rysunek 10: Przykłady dopasowania w regułach diagnostycznych

Implementacja klauzuli *problem_cause*:

problem_cause(*Problem*) :-

solution(*problem*(*Problem*), *applies_to*(*Applies*), *Solution*), *print_sol*(*Applies*, *Solution*).

```

solution(problem(brew_unit_cannot_be_removed), applies_to(brew_unit), _).
true
solution(problem(brew_unit_cannot_be_removed), applies_to(brew_unit), Solution).
Solution = 'Close the service door. Disable\n device and turn it back on.\n Wait until appears on the display\n "Machine ready" message, then remove the unit\n brewing.'
solution(problem(brew_unit_cannot_be_removed), applies_to( Applies ), Solution).
Applies = brew_unit,
Solution = 'Close the service door. Disable\n device and turn it back on.\n Wait until appears on the display\n "Machine ready" message, then remove the unit\n brewing.'
solution(problem(too_cold), applies_to( Applies ), _).
Applies = dispenser
solution(problem(too_cold), _, Solution).
Solution = 'Clean the coffee spout and its spout\n\t holes with a ramrod.'

```

Rysunek 11: Inne przykłady dopasowania w regułach diagnostycznych

5 Dodatek: uruchamianie ekspresu

Ciekawą możliwością, jaką dostarcza nam język Prolog, jest inicjalizacja dynamicznych atomów. Za pomocą klauzuli *retract* możemy usuwać własności do nich przypisane, a za pomocą klauzuli *assert* dodawać nowe. Posłuży nam to do zdefiniowania działań wymaganych podczas uruchamiania maszyny. Przyjrzyjmy się temu mechanizmowi na poniższym przykładzie:

```

%Basic usage rules

:- dynamic machine_on/1.
:- dynamic water_level/1.
:- dynamic coffee_powder/1.
:- dynamic cup_placed/1.

machine_on(off).
water_level(low).
coffee_powder(empty).
cup_placed(no).

turn_on_machine :-
    machine_on(off),
    retract(machine_on(off)),
    assert(machine_on(on)),
    write('The coffee machine is now on.').

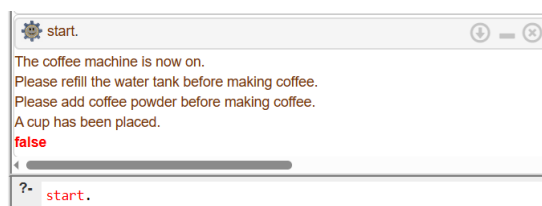
turn_off_machine :-
    machine_on(on),
    retract(machine_on(on)),
    assert(machine_on(off)),
    write('The coffee machine is now off.').

make_coffee :-
    machine_on(on),
    water_level(high),
    coffee_powder(not_empty),
    cup_placed(yes),
    write('Making coffee... Enjoy!').

% Usage example
start :-
    turn_on_machine,nl,
    check_water_level,nl,
    check_coffee_powder,nl,
    place_cup,nl,
    make_coffee,nl,
    remove_cup,nl,
    turn_off_machine.

```

Rysunek 12: Dynamicznie zdefiniowane własności maszyny oraz klauzule do ich modyfikacji i uruchomienia maszyny



```

start.
The coffee machine is now on.
Please refill the water tank before making coffee.
Please add coffee powder before making coffee.
A cup has been placed.
false
?- start.

```

Rysunek 13: Uruchomienie klauzuli start