Lab3 - Malakser

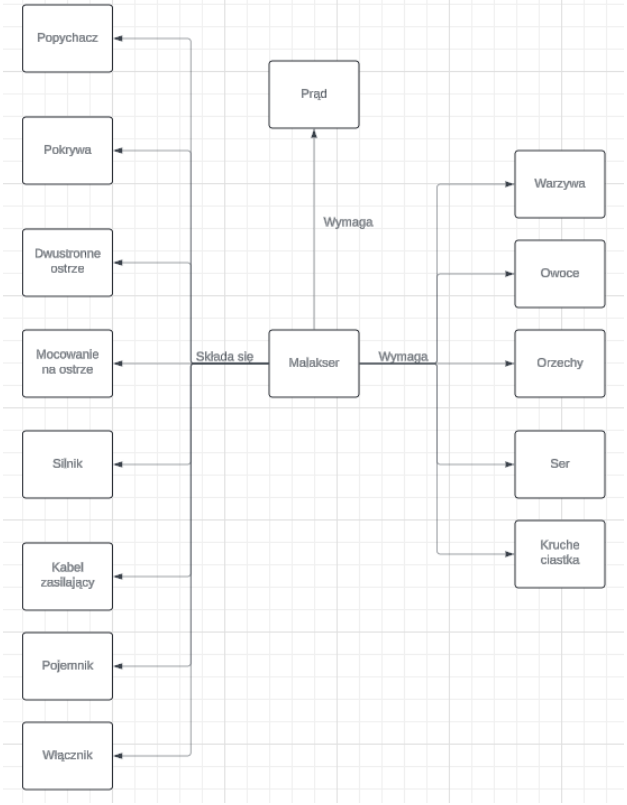
Jakub Ner (266543)

Wprowadzenie

Celem ćwiczenia jest zaprezentowanie wiedzy dotyczącej malaksera firmy Kenwood w postaci faktów i reguł zapisanych w reprezentacji symbolicznej.

Sieć semantyczna

Dla Malaksera stworzyłem sieć semantyczną opisującą elementy składowe i wymagane do jego prawidłowego działania.



Powstała sieć przełożyła się na stworzenie poniższych faktów:

```
class Malakser(Fact):pass
class MalakserKomponent(Fact):pass
class PrzygotowanyMalakser(Fact):pass

class Skladnik(Fact):pass

class Problem(Fact):pass
class Przyczyna(Fact):pass
class Rozwaizanie(Fact):pass

class STRONA_OSTRZA:
    TNACA = "tnąca na talarki"
    TRACA = "trąca na wiórki"

class ZASILANIE:
    BRAK = "brak prądu"
    JEST = "jest prąd"
```

Z malaksera można skorzystać dopiero po umieszczeniu składników w pojemniku malaksera, prawidłowym zmontowaniu urządzenia, podłączeniu do prądu i uruchomieniu go:

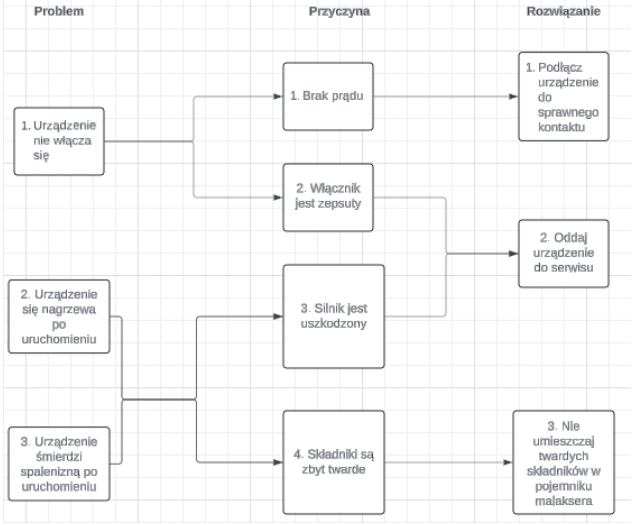
```
przygotuj_salatke_z_uzyciem_popychacza():
print("\nPrzygotuj sałatkę z użyciem popychacza:\n")
engine.reset()

# Zmontuj malakser
engine.declare(Malakser())
engine.declare(
MalakserKomponent(nazwa="pokrywa"),
MalakserKomponent(nazwa="mocowanie na ostrze"),
MalakserKomponent(nazwa="pojemnik"),
MalakserKomponent(nazwa="ostrze dwustronne", strona=STRONA_OSTRZA.TNACA),
)
# przygotuj sałatkę
engine.declare(
    Skladnik(nazwa={"owoce", "warzywa"}),
    MalakserKomponent(nazwa="kabel zasilajacy", stan=ZASILANIE.JEST),
    MalakserKomponent(nazwa="włącznik", wlaczony=True),
    MalakserKomponent(nazwa="popychacz"),
)
engine.run()
```

```
jakubner:ZAD3 baza wiedzy i wnioskowanie$
jakubner:ZAD3 baza wiedzy i wnioskowanie$
jakubner:ZAD3 baza wiedzy i wnioskowanie$
jakubner:ZAD3 baza wiedzy i wnioskowanie$
jakubner:ZAD3 baza wiedzy i wnioskowanie$python main.py

Przygotuj sałatkę z użyciem popychacza:

Zamontowano ostrze tnąca na talarki
Złożono malakser
Zawartość blendera: warzywa, owoce, starta na talarki z użyciem popychacza
-----------------------------------

jakubner:ZAD3 baza wiedzy i wnioskowanie$
```

```
class MalakserExpert(KnowledgeEngine):
    @DefFacts()
    def komponenty_scalone(self):
        yield MalakserKomponent(nazwa="silnik")
        yield MalakserKomponent(nazwa="kabel zasilajacy", stan=ZASILANIE.BRAK)
        yield MalakserKomponent(nazwa="włącznik", wlaczony=False)
        yield PrzygotowanyMalakser(nazwa="nie")

    @Rule(AND(
        Malakser(),
        PrzygotowanyMalakser(nazwa="tak"),
        MalakserKomponent(nazwa="kabel zasilajacy", stan=ZASILANIE.JEST),
        MalakserKomponent(nazwa="włącznik", wlaczony=True),
        MalakserKomponent(nazwa="ostrze dwustronne", strona=MATCH.strona),
        Skladnik(nazwa=MATCH.nazwa),
        OR(
            MalakserKomponent(nazwa="popychacz"),
            NOT(MalakserKomponent(nazwa="popychacz")),
        ),
    ))
    def uzyj_malaksera(self, nazwa, strona):
        with_popychacz = [fact["nazwa"] for fact in self.facts.values() if isinstance(fact, MalakserKomponent)]
        skladniki = ", ".join(nazwa)
        if "popychacz" in with_popychacz:
            print(f"Zawartość blendera: {skladniki}, starta na {strona.split(' ')[-1]} z użyciem popychacza")
        else:
            print(f"Zawartość blendera: {skladniki}, starta na surówkę")

    @Rule(
        Malakser(),
        MalakserKomponent(nazwa="pokrywa"),
        MalakserKomponent(nazwa="ostrze dwustronne"),
        MalakserKomponent(nazwa="mocowanie na ostrze"),
        MalakserKomponent(nazwa="pojemnik"),
        salience=1)
    def zloz_malakser(self):
        self.declare(PrzygotowanyMalakser(nazwa="tak"))
        print("Złożono malakser")

    @Rule(
        Malakser(),
        MalakserKomponent(nazwa="pokrywa"),
        MalakserKomponent(nazwa="ostrze dwustronne", strona=MATCH.strona),
        MalakserKomponent(nazwa="mocowanie na ostrze"),
        salience=2)
    def zamontuj_ostrze(self, strona):
        print("Zamontowano ostrze " + strona)
```

Diagram rozwiązywania problemów

Zdefiniowałem drzewo możliwych problemów z urządzeniem, potencjalnych przyczyny oraz rozwiązań.



Dzięki temu dla zadanego problemu mogę znaleźć przyczyny i rozwiązania, a dla podanych przyczyn - rozwiązania.

```
def znajdz_rozwiazanie_problemu(problem):
    print("\nSzukam rozwiązań dla problemu: " + problem + ':\n')

    engine.reset()
    engine.declare(Problem(nazwa=problem))
    engine.run()
    print("-----------------------------------")


znajdz_rozwiazanie_problemu(PROBLEM1)
znajdz_rozwiazanie_problemu(PROBLEM2)
znajdz_rozwiazanie_problemu(PROBLEM3)
```

```
Szukam rozwiązań dla problemu: Urządzenie nie włącza się:

Wystąpił problem: Urządzenie nie włącza się
  Potencjalna przyczyna: Włącznik jest zepsuty
    Rozwiązanie: Oddaj urządzenie do serwisu
  Potencjalna przyczyna: Brak prądu
    Rozwiązanie: Podłącz urządzenie do sprawnego kontaktu
-----------------------------------

Szukam rozwiązań dla problemu: Urządzenie się nagrzewa po uruchomieniu:

Wystąpił problem: Urządzenie się nagrzewa po uruchomieniu
  Potencjalna przyczyna: Składniki są zbyt twarde
    Rozwiązanie: Nie umieszczaj twardych składników w pojemniku malaksera
  Potencjalna przyczyna: Silnik jest uszkodzony
    Rozwiązanie: Oddaj urządzenie do serwisu
-----------------------------------

Szukam rozwiązań dla problemu: Urządzenie śmierdzi spalenizną po uruchomieniu:

Wystąpił problem: Urządzenie śmierdzi spalenizną po uruchomieniu
  Potencjalna przyczyna: Składniki są zbyt twarde
    Rozwiązanie: Nie umieszczaj twardych składników w pojemniku malaksera
  Potencjalna przyczyna: Silnik jest uszkodzony
    Rozwiązanie: Oddaj urządzenie do serwisu
-----------------------------------
```

```
#### Rozwiązania:
@Rule(Rozwaizanie(nazwa=ROZWIAZANIE1))
def rozwiazanie1(self):
    print("    Rozwiązanie: " + ROZWIAZANIE1)

@Rule(Rozwaizanie(nazwa=ROZWIAZANIE2))
def rozwiazanie2(self):
    print("    Rozwiązanie: " + ROZWIAZANIE2)

@Rule(Rozwaizanie(nazwa=ROZWIAZANIE3))
def rozwiazanie3(self):
    print("    Rozwiązanie: " + ROZWIAZANIE3)

#### Przyczyny:
@Rule(Przyczyna(nazwa=PRZYCZYNA1))
def przyczyna1(self):
    print("  Potencjalna przyczyna: " + PRZYCZYNA1)
    self.declare(Rozwaizanie(nazwa=ROZWIAZANIE1))

@Rule(Przyczyna(nazwa=PRZYCZYNA2))
def przyczyna2(self):
    print("  Potencjalna przyczyna: " + PRZYCZYNA2)
    self.declare(Rozwaizanie(nazwa=ROZWIAZANIE2))

@Rule(Przyczyna(nazwa=PRZYCZYNA3))
def przyczyna3(self):
    print("  Potencjalna przyczyna: " + PRZYCZYNA3)
    self.declare(Rozwaizanie(nazwa=ROZWIAZANIE2))

@Rule(Przyczyna(nazwa=PRZYCZYNA4))
def przyczyna4(self):
    print("  Potencjalna przyczyna: " + PRZYCZYNA4)
    self.declare(Rozwaizanie(nazwa=ROZWIAZANIE3))

#### PROBLEMY:
@Rule(Problem(nazwa=PROBLEM1))
def problem1(self):
    print("Wystąpił problem: " + PROBLEM1)
    self.declare(Przyczyna(nazwa=PRZYCZYNA1))
    self.declare(Przyczyna(nazwa=PRZYCZYNA2))

@Rule(Problem(nazwa=PROBLEM2))
def problem2(self):
    print("Wystąpił problem: " + PROBLEM2)
    self.declare(Przyczyna(nazwa=PRZYCZYNA3))
    self.declare(Przyczyna(nazwa=PRZYCZYNA4))
```

Wnioski:

Wykorzystanie bazy wiedzy pozwala na opisanie możliwych akcji urządzenia, a uruchamianie procesu wnioskowania w przód umożliwia skuteczne diagnozowanie problemów i proponowanie odpowiednich rozwiązań.