Poglavlje 2

Wittgensteinova logička slika svijeta

2.1 Semantička logička posljedica kroz prizmu *Tractatusa*

U ovom poglavlju istražujemo koncept **semantičke logičke posljedice** kroz praktičnu Python implementaciju, inspirirani Wittgensteinovim pristupom logici i ontologiji iz *Tractatus Logico-Philosophicus*.

> "Svijet je sve što je slučaj" (*Die Welt ist alles, was der Fall ist*) - Tractatus, 1

Ova slavna prva rečenica *Tractatusa* postavlja temelje za razumijevanje odnosa između jezika, logike i stvarnosti.

2.1.1 Atomarne činjenice i elementarni sudovi

Za Wittgensteina, svijet se sastoji od **činjenica** (*Tatsachen*), ne stvari:

> "Svijet je totalitet činjenica, ne stvari" - Tractatus, 1.1

Atomarne činjenice (*Sachverhalte*) su najjednostavnije činjenice koje mogu postojati ili ne postojati. U logici ih predstavljamo **elementarnim sudovima**.

```python
class ElementarniSud:
    """Predstavlja atomarnu činjenicu u Wittgensteinovom smislu."""

    def __init__(self, simbol, opis=""):
        self.simbol = simbol
        self.opis = opis
        self.vrijednost = None
```

```

    def __repr__(self):
        return f"{self.simbol}: {self.opis}"

    def __bool__(self):
        return bool(self.vrijednost)

# Stvorimo elementarne sudove koji odgovaraju atomarnim činjenicama
p = ElementarniSud("p", "Pada kiša")
q = ElementarniSud("q", "Ulice su mokre")
r = ElementarniSud("r", "Nosim kišobran")

print("Elementarni sudovi (atomarne činjenice):")
print(f"  {p}")
print(f"  {q}")
print(f"  {r}")
```

Output

```
Elementarni sudovi (atomarne činjenice):
  p: Pada kiša
  q: Ulice su mokre
  r: Nosim kišobran
```

2.1.2 Logički prostor i mogući svjetovi

Wittgenstein uvodi pojam **logičkog prostora** kao totaliteta svih mogućih konfiguracija atomarnih činjenica:

> "Činjenice u logičkom prostoru jesu svijet" - Tractatus, 1.13

Svaki **mogući svijet** je jedna određena kombinacija postojanja i nepostojanja atomarnih činjenica.

```
import itertools

def generiraj_moguce_svjetove(elementarni_sudovi):
    """Generira sve moguće svjetove kao kombinacije istinitosnih vrijednosti."""
    svjetovi = []
    n = len(elementarni_sudovi)

    for vrijednosti in itertools.product([False, True], repeat=n):
        svijet = {}
        for sud, vrijednost in zip(elementarni_sudovi, vrijednosti):
            svijet[sud.simbol] = vrijednost
        svjetovi.append(svijet)
```

```
13
14      return svjetovi
15
16  # Generiraj logički prostor
17  sudovi = [p, q, r]
18  svi_svjetovi = generiraj_moguce_svjetove(sudovi)
19
20  print(f"Logički prostor sadrži {len(svi_svjetovi)} mogućih svjetova:\n")
21  for i, svijet in enumerate(svi_svjetovi[:4], 1):  # Prikaži prve 4
22      # Koristi simbole ⊤ i ⊥ umjesto True/False
23      svijet_prikaz = {k: '⊤' if v else '⊥' for k, v in svijet.items()}
24      print(f"Svijet {i}: {svijet_prikaz}")
25  print("...")
```

Output

```
Logički prostor sadrži 8 mogućih svjetova:

Svijet 1: {'p': '⊥', 'q': '⊥', 'r': '⊥'}
Svijet 2: {'p': '⊥', 'q': '⊥', 'r': '⊤'}
Svijet 3: {'p': '⊥', 'q': '⊤', 'r': '⊥'}
Svijet 4: {'p': '⊥', 'q': '⊤', 'r': '⊤'}
...
```

2.1.3 Logički veznici i složeni sudovi

Wittgenstein pokazuje kako se složeni sudovi grade iz elementarnih pomoću **istinosno-funkcionalnih veznika**:

> "Sud je istinosna funkcija elementarnih sudova" - Tractatus, 5

Implementirajmo osnovne logičke veznike koristeći Python operatore:

```
1   class Sud:
2       """Predstavlja sud koji može biti elementaran ili složen."""
3
4       def __init__(self, formula, opis=""):
5           self.formula = formula
6           self.opis = opis
7           self.evaluacija = None
8
9       def evaluiraj(self, svijet):
10          """Evaluira sud u danom mogućem svijetu."""
11          # Ovdje bi trebala biti logika evaluacije
12          # Za sada vraćamo jednostavnu vrijednost
13          return self.evaluacija if self.evaluacija is not None else False
```

```python

    def __repr__(self):
        return self.formula

# Definiraj funkcije za logičke veznike
def negacija(sud):
    """Negacija: ¬p"""
    return Sud(f"¬{sud.formula}", f"nije slučaj da {sud.opis}")

def konjunkcija(sud1, sud2):
    """Konjunkcija: p ∧ q"""
    return Sud(f"({sud1.formula} ∧ {sud2.formula})",
               f"{sud1.opis} i {sud2.opis}")

def disjunkcija(sud1, sud2):
    """Disjunkcija: p ∨ q"""
    return Sud(f"({sud1.formula} ∨ {sud2.formula})",
               f"{sud1.opis} ili {sud2.opis}")

def implikacija(sud1, sud2):
    """Materijalna implikacija: p → q"""
    return Sud(f"({sud1.formula} → {sud2.formula})",
               f"ako {sud1.opis}, onda {sud2.opis}")

# Primjeri složenih sudova
p_sud = Sud("p", "pada kiša")
q_sud = Sud("q", "ulice su mokre")

složeni1 = implikacija(p_sud, q_sud)
složeni2 = konjunkcija(p_sud, negacija(q_sud))

print("Složeni sudovi:")
print(f"  {složeni1}: {složeni1.opis}")
print(f"  {složeni2}: {složeni2.opis}")
```

Output

```
Složeni sudovi:
  (p → q): ako pada kiša, onda ulice su mokre
  (p ∧ ¬q): pada kiša i nije slučaj da ulice su mokre
```

2.1.4 Semantička logička posljedica

Ključni koncept u logici je **logička posljedica** (semantička implikacija). Kažemo da je sud φ logička posljedica skupa sudova Γ, što zapisujemo:

$$\Gamma \models \varphi$$

ako i samo ako u **svakom mogućem svijetu** gdje su svi sudovi iz Γ istiniti, φ je također istinit.

Za Wittgensteina, logička posljedica pokazuje strukturalni odnos između sudova:

> "Kada istinitost jednog suda slijedi iz istinitosti drugih, to vidimo iz strukture samih sudova" - Tractatus, 5.13

```python
def je_logicka_posljedica(premisa, zakljucak, elementarni):
    """
    Provjerava je li zaključak logička posljedica premise.

    Args:
        premisa: funkcija koja evaluira premisu
        zakljucak: funkcija koja evaluira zaključak
        elementarni: lista elementarnih sudova

    Returns:
        (bool, protuprimjer ili None)
    """
    for vrijednosti in itertools.product([False, True], repeat=len(elementarni)):
        # Postavi vrijednosti elementarnih sudova
        svijet = dict(zip(elementarni, vrijednosti))

        # Evaluiraj premisu i zaključak
        p_istina = premisa(svijet)
        z_istina = zakljucak(svijet)

        # Ako premisa istinita a zaključak lažan - nije logička posljedica
        if p_istina and not z_istina:
            return False, svijet

    return True, None

# Definiraj jednostavne evaluacijske funkcije
def p_eval(svijet):
    return svijet.get('p', False)

def p_imp_q(svijet):
    p = svijet.get('p', False)
    q = svijet.get('q', False)
    return not p or q  # p → q ⇔ ¬p ∨ q

def q_eval(svijet):
    return svijet.get('q', False)

# Test: Je li q logička posljedica od (p → q) ∧ p? (Modus Ponens)
def modus_ponens_premisa(svijet):
    return p_imp_q(svijet) and p_eval(svijet)

```

```
rezultat, protuprimjer = je_logicka_posljedica(
    modus_ponens_premisa,
    q_eval,
    ['p', 'q']
)

print("Test Modus Ponens: ((p → q) ∧ p) ⊨ q")
if rezultat:
    print("  ✓ JEST logička posljedica")
else:
    print(f"  × NIJE logička posljedica. Protuprimjer: {protuprimjer}")
```

Output

```
Test Modus Ponens: ((p → q) ∧ p) ⊨ q
  ✓ JEST logička posljedica
```

2.1.5 Tautologije i kontradikcije

Wittgenstein ističe poseban status **tautologija** i **kontradikcija**:

> "Tautologija i kontradikcija nisu slike stvarnosti. One ne predstavljaju nikakvu moguću situaciju" - Tractatus, 4.462

- **Tautologija**: istinita u svim mogućim svjetovima (npr. $p \vee \neg p$)
- **Kontradikcija**: lažna u svim mogućim svjetovima (npr. $p \wedge \neg p$)

One pokazuju granice logičkog prostora:

```
def provjeri_status(formula_eval, elementarni):
    """Provjerava je li formula tautologija, kontradikcija ili kontingentna."""
    istiniti = 0
    ukupno = 0

    for vrijednosti in itertools.product([False, True], repeat=len(elementarni)):
        svijet = dict(zip(elementarni, vrijednosti))
        if formula_eval(svijet):
            istiniti += 1
        ukupno += 1

    if istiniti == ukupno:
        return "TAUTOLOGIJA"
    elif istiniti == 0:
        return "KONTRADIKCIJA"
```

```
    else:
        return f"KONTINGENTNA ({istiniti}/{ukupno} svjetova)"

# Testiraj različite formule
def tautologija(svijet):
    p = svijet.get('p', False)
    return p or not p  # p ∨ ¬p

def kontradikcija(svijet):
    p = svijet.get('p', False)
    return p and not p  # p ∧ ¬p

def kontingentna(svijet):
    return svijet.get('p', False)  # samo p

print("Status formula u logičkom prostoru:\n")
print(f"p ∨ ¬p: {provjeri_status(tautologija, ['p'])}")
print(f"p ∧ ¬p: {provjeri_status(kontradikcija, ['p'])}")
print(f"p: {provjeri_status(kontingentna, ['p'])}")
```

Output

```
Status formula u logičkom prostoru:

p ∨ ¬p: TAUTOLOGIJA
p ∧ ¬p: KONTRADIKCIJA
p: KONTINGENTNA (1/2 svjetova)
```

2.1.6 Tablični prikaz logičkih posljedica

Vizualizirajmo logičke posljedice pomoću **tablica istinitosti**, što odgovara Wittgensteinovoj metodi iz *Tractatusa*:

```
def tablica_istinitosti(premise, zakljucak, varijable):
    """Generira tablicu istinitosti za provjeru logičke posljedice."""
    print("\nTablica istinitosti:")
    print("=" * 60)

    # Zaglavlje
    header = " | ".join(varijable) + " | Premisa | Zaključak | Status"
    print(header)
    print("-" * len(header))

    je_posljedica = True

    for vrijednosti in itertools.product([" ⊥ ", " ⊤ "], repeat=len(varijable)):
        svijet = {var: (val.strip() == "⊤") for var, val in zip(varijable, vrijednosti)}

```

```
        p_val = premise(svijet)
        z_val = zakljucak(svijet)

        # Provjeri je li narušena logička posljedica
        status = ""
        if p_val and not z_val:
            status = "<-- PROTUPRIMJER"
            je_posljedica = False

        # Ispis reda
        row = " | ".join(vrijednosti)
        row += f" |    {'⊤' if p_val else '⊥'}    |    {'⊤' if z_val else '⊥'}    |
        ↪ {status}"
        print(row)

    print("=" * 60)
    if je_posljedica:
        print("\n✓ Zaključak JEST logička posljedica premise")
    else:
        print("\n× Zaključak NIJE logička posljedica premise")

    return je_posljedica

# Test klasičnih logičkih zakona
print("\nMODUS PONENS: ((p → q) ∧ p) ⊨ q")
tablica_istinitosti(modus_ponens_premisa, q_eval, ['p', 'q'])

# Test disjunktivnog silogizma
def disj_silogizam_premisa(svijet):
    p = svijet.get('p', False)
    q = svijet.get('q', False)
    return (p or q) and not p

print("\nDISJUNKTIVNI SILOGIZAM: ((p ∨ q) ∧ ¬p) ⊨ q")
tablica_istinitosti(disj_silogizam_premisa, q_eval, ['p', 'q'])
```

Output

```
MODUS PONENS: ((p → q) ∧ p) ⊨ q

Tablica istinitosti:
============================================================
p | q | Premisa | Zaključak | Status
------------------------------------
 ⊥  |  ⊥  |    ⊥    |    ⊥    |
 ⊥  |  ⊤  |    ⊥    |    ⊤    |
 ⊤  |  ⊥  |    ⊥    |    ⊥    |
 ⊤  |  ⊤  |    ⊤    |    ⊤    |
============================================================

✓ Zaključak JEST logička posljedica premise
```

```
DISJUNKTIVNI SILOGIZAM: ((p ∨ q) ∧ ¬p) ⊨ q

Tablica istinitosti:
============================================================
p | q | Premisa | Zaključak | Status
------------------------------------
 ⊥  |  ⊥  |    ⊥    |     ⊥     |
 ⊥  |  ⊤  |    ⊤    |     ⊤     |
 ⊤  |  ⊥  |    ⊥    |     ⊥     |
 ⊤  |  ⊤  |    ⊥    |     ⊤     |
============================================================

✓ Zaključak JEST logička posljedica premise
True
```

2.1.7 Filozofske implikacije

Granice jezika i logike

Wittgenstein pokazuje da logika ima svoje granice:

> "Logika ispunjava svijet; granice svijeta su također njezine granice" - Tractatus, 5.61

Naša implementacija demonstrira ove granice:

- **Tautologije** ne govore ništa o svijetu (istinite su uvijek)
- **Kontradikcije** opisuju nemogućnosti
- Samo **kontingentne formule** zapravo opisuju moguća stanja svijeta

Slikovna teorija značenja

Prema Wittgensteinu, sudovi su **slike mogućih stanja stvari**:

> "Logička slika činjenica jest misao" - Tractatus, 3

Naš kod modelira ovu ideju:

- Elementarni sudovi = atomarne činjenice
- Složeni sudovi = kombinacije atomarnih činjenica
- Mogući svjetovi = sve moguće konfiguracije

2.1.8 Praktična primjena: Logičko zaključivanje

Implementirajmo jednostavan sustav za automatsko logičko zaključivanje:

```python
class LogickiSustav:
    """Jednostavan sustav za logičko zaključivanje."""

    def __init__(self):
        self.baza_znanja = []
        self.elementarni = set()

    def dodaj_premisu(self, formula, varijable):
        """Dodaje premisu u bazu znanja."""
        self.baza_znanja.append(formula)
        self.elementarni.update(varijable)

    def moze_zakljuciti(self, zakljucak):
        """Provjerava slijedi li zaključak iz baze znanja."""

        def premise_eval(svijet):
            # Sve premise moraju biti istinite
            for premisa in self.baza_znanja:
                if not premisa(svijet):
                    return False
            return True

        # Provjeri sve moguće svjetove
        for vrijednosti in itertools.product([False, True],
                                             repeat=len(self.elementarni)):
            svijet = dict(zip(list(self.elementarni), vrijednosti))

            if premise_eval(svijet) and not zakljucak(svijet):
                return False, svijet  # Našli protuprimjer

        return True, None

# Primjer korištenja
sustav = LogickiSustav()

# Dodaj premise
def ako_kisa_mokro(svijet):
    return not svijet.get('kiša', False) or svijet.get('mokro', False)

def kisa_pada(svijet):
    return svijet.get('kiša', False)

sustav.dodaj_premisu(ako_kisa_mokro, ['kiša', 'mokro'])
sustav.dodaj_premisu(kisa_pada, ['kiša'])

# Testiraj zaključke
def ulice_mokre(svijet):
```

```
    return svijet.get('mokro', False)

rezultat, protuprimjer = sustav.moze_zakljuciti(ulice_mokre)

print("Baza znanja:")
print("  1. Ako pada kiša, ulice su mokre")
print("  2. Pada kiša")
print("\nZaključak: Ulice su mokre")
print(f"\nRezultat: {'✓ Logički slijedi' if rezultat else '× Ne slijedi'}")
```

2.1.9 Zadaci i prijedlozi za daljnje istraživanje

Za produbljivanje razumijevanja semantičke logičke posljedice i Wittgensteinove filozofije, predlažemo sljedeće istraživačke teme prikladne za dodiplomske studente:

2.1.10 Prijedlozi za daljnje istraživanje

Za produbljivanje razumijevanja semantičke logičke posljedice kroz praktične Python zadatke, predlažemo sljedeće vježbe prikladne za studente koji uče osnove logike sudova:

Proširenje skupa logičkih veznika

Implementirajte funkcije za dodatne logičke veznike: ekskluzivnu disjunkciju (XOR), Shefferovu crticu (NAND) i Pierceovu strelicu (NOR). Pokažite da su NAND i NOR funkcionalno potpuni - da pomoću njih možete izraziti sve ostale veznike.

Automatska generacija tablica istinitosti

Napišite funkciju koja prima proizvoljan logički izraz kao string (npr. "(p → q) ∧ (q → r)") i automatski generira njegovu tablicu istinitosti. Koristite Python `eval()` funkciju uz sigurnosne provjere.

Prepoznavanje tautologija

Stvorite funkciju koja provjerava je li dana formula tautologija bez generiranja cijele tablice istinitosti - zaustavite se čim nađete protuprimjer. Testirajte na klasičnim zakonima: De Morganovim zakonima, zakonu distribucije, zakonu kontrapozicije.

Normalne forme

Implementirajte pretvorbu formula u konjunktivnu (KNF) i disjunktivnu (DNF) normalnu formu. Za danu tablicu istinitosti generirajte minimalnu formulu koja je opisuje.

Provjera ekvivalencije

Napišite funkciju koja provjerava jesu li dvije formule logički ekvivalentne. Testirajte s primjerima poput: je li $(p \to q)$ ekvivalentno s $(\neg p \vee q)$? Je li $(p \to (q \to r))$ ekvivalentno s $((p \wedge q) \to r)$?

Broj mogućih formula

Za n propozicijskih varijabli, koliko različitih (neekvivalentnih) formula možete stvoriti? Napišite program koji generira sve moguće tablice istinitosti za 2 i 3 varijable i broji koliko ih je jedinstvenih.

Vizualizacija logičkih odnosa

Koristeći matplotlib, nacrtajte graf gdje čvorovi predstavljaju moguće svjetove, a bridovi povezuju svjetove koji se razlikuju u točno jednoj atomarnoj činjenici. Obojite svjetove gdje je vaša formula istinita.

Minimalni skup premisa

Za dani zaključak i skup premisa, pronađite minimalni podskup premisa iz kojeg zaključak još uvijek slijedi. Na primjer, ako imate premise $p, p \to q, q \to r, p \to r$, koji je minimalni skup za zaključak r?

Interaktivni dokazivač

Stvorite jednostavnu interaktivnu aplikaciju gdje korisnik može graditi dokaz korak po korak koristeći osnovna pravila (modus ponens, modus tollens, disjunktivni silogizam). Program provjerava valjanost svakog koraka.

Analiza složenosti formula

Napišite funkcije koje mjere "složenost" formule: broj veznika, dubinu ugniježđenja, broj različitih varijabli. Istražite odnos između složenosti formule i broja redaka u njenoj minimalnoj DNF reprezentaciji.

Svaki zadatak postupno gradi razumijevanje ključnih koncepata semantičke logičke posljedice kroz praktično programiranje, omogućavajući studentima da eksperimentiraju s logičkim strukturama i razviju intuiciju za formalno zaključivanje.

2.1.11 Zaključak

Kroz ovu implementaciju istražili smo temeljne koncepte semantičke logičke posljedice inspirirani Wittgensteinovom filozofijom:

1. **Atomarne činjenice** kao građevni blokovi stvarnosti
2. **Logički prostor** kao totalitet mogućih svjetova
3. **Logička posljedica** kao odnos koji vrijedi u svim mogućim svjetovima
4. **Granice logike** pokazane kroz tautologije i kontradikcije

Wittgensteinov zaključak *Tractatusa* podsjeća nas:

> "O čemu se ne može govoriti, o tome se mora šutjeti" - Tractatus, 7

Logika može opisati strukturu mogućih svjetova, ali sama ta sposobnost počiva na metalogičkim osnovama koje se ne mogu izraziti unutar sustava. Naš Python kod demonstrira ovu granicu - možemo implementirati logiku, ali pitanje zašto logika funkcionira ostaje izvan dosega same logike.

Kroz praktično programiranje otkrivamo da je razumijevanje logičke posljedice ključno ne samo za filozofiju jezika i logike, već i za moderna područja poput verifikacije softvera, umjetne inteligencije i automatskog zaključivanja.