Poglavlje 1

Uvod: Što je logika i zašto kod?

> Ne priliči izvrsnim ljudima da kao robovi gube sate na računanje, u poslu koji se može s punim povjerenjem prepustiti bilo kome drugome uporabom strojeva.
>
> Gottfried Wilhelm Leibniz[1]

1.1 Logika kao znanost o valjanom zaključivanju

Logika proučava oblike valjanog zaključivanja i relacije logičkog slijeda. Kada kažemo da je zaključak valjan, mislimo da konkluzija nužno slijedi iz premisa. Ova nužnost proizlazi iz same strukture našeg mišljenja, neovisno o sadržaju.

Razmotrimo klasičan primjer:

1. Svi ljudi su smrtni.
2. Sokrat je čovjek.
3. Dakle, Sokrat je smrtan.

Valjanost ovog zaključka ne ovisi o tome tko je Sokrat ili što znači čovjek" i smrtan". Ona proizlazi iz logičke forme koja ostaje valjana čak i kad zamijenimo termine:

1. Svi P su Q.
2. a je P.
3. Dakle, a je Q.

[1] *Machina Arithmetica in qua non Aditio tantum Subtractio 1685*, slobodni prijevod s engleskog prijevoda

Moderna simbolička logika omogućava nam precizno zapisivanje ovakvih formi. U logici sudova bavimo se veznicima:

- Konjunkcija ($\wedge$): „i”
- Disjunkcija ($\vee$): „ili”
- Implikacija ($\rightarrow$): „ako...onda”
- Negacija ($\neg$): „nije”

Logika predikata ide dalje omogućavajući kvantifikaciju:

- Univerzalni kvantifikator ($\forall$): „svi”, „svaki”
- Egzistencijalni kvantifikator ($\exists$): „neki”, „postoji”

1.2 Formalni jezik misli

Gottlob Frege, utemeljitelj moderne logike, nastojao je stvoriti ..pojmovno pismo” (*Begriffsschrift*) – formalni jezik za izražavanje čistog mišljenja, oslobođenog dvosmislenosti prirodnog jezika. Njegova vizija danas živi u programskim jezicima.

Kada pišemo Python kod:

```
def je_smrtan(x):
    return je_čovjek(x)

assert je_smrtan("Sokrat") == True
```

formaliziramo logičku strukturu. Funkcija `je_smrtan` enkodira univerzalnu tvrdnju, a `assert` provjerava konkretnu instancu.

1.3 Struktura stvarnosti: Wittgensteinova slika

Ludwig Wittgenstein u *Tractatus Logico-Philosophicus* nudi radikalnu tezu: granice našeg jezika su granice našeg svijeta. Za njega, svijet je totalitet činjenica, ne stvari. Činjenice postoje u ..logičkom prostoru” – prostoru svih mogućih kombinacija.

Ova ideja ima izravnu programsku interpretaciju. Kada definiramo Booleove varijable:

```
kiša = True
sunce = False
vjetar = True
```

definiramo točku u logičkom prostoru. S tri varijable, imamo $2^3 = 8$ mogućih svjetova. Svaki program implicitno definira takav prostor mogućnosti i pravila kretanja kroz njega.

1.4 Istina i značenje: Tarskijeva semantika

Alfred Tarski riješio je drevni problem istine kroz svoju semantičku teoriju. Njegova čuvena T-shema:

..Snijeg je bijel" je istinito ako i samo ako snijeg je bijel.

Može se činiti trivijalnom, ali razlikuje jezik od metajezika. U Pythonu:

```
def je_istinito(izjava, model):
    return eval(izjava, model)

model = {"snijeg_je_bijel": True}
assert je_istinito("snijeg_je_bijel", model) == True
```

funkcija `je_istinito` je metajezična – ona govori *o* izjavama, ne *u* njima.

1.5 Dokazi kao programi: Curry-Howard izomorfizam

Jedan od najdubljih uvida 20. stoljeća je otkriće strukturne ekvivalencije između logičkih dokaza i programa. Curry-Howard izomorfizam pokazuje:

Logika	**Programiranje**
Formula $A \to B$	Tip funkcije `A -> B`
Dokaz formule	Program tog tipa
Modus ponens	Aplikacija funkcije
Konjunkcija $A \wedge B$	Tuple `(A, B)`
Disjunkcija $A \vee B$	Union type `A | B`

Svaki put kad pišete funkciju, vi zapravo konstruirate dokaz. Svaki put kad je pozivate, primjenjujete logičko pravilo. Tipovi su teoremi, programi su dokazi!

1.6 Granice formalizma: Gödelova nepotpunost

Kurt Gödel je 1931. dokazao da svaki dovoljno bogat formalni sustav ili je nekonzistentan ili nepotpun. Postoje istinite tvrdnje koje se ne mogu dokazati unutar sustava.

Njegov dokaz koristi samoreferenciranje – konstruira rečenicu koja kaže ..Ja nisam dokaziva". Ako je dokaziva, sustav je nekonzistentan. Ako nije, sustav je nepotpun.

U Pythonu možemo ilustrirati Gödelov trik:

```
def gödel_rečenica(n):
```

```
    """Rečenica koja tvrdi da nije dokaziva"""
    return f"Rečenica broj {n} nije dokaziva"


\textbf{Paradoks: ako je dokaziva, onda je lažna}

\textbf{Ako nije dokaziva, onda je istinita, ali nedokaziva}
```

1.7 Turingovi strojevi

Alan Turing definirao je precizni model računanja – Turingov stroj. Pokazao je da postoje problemi koje nijedan stroj ne može riješiti, poput problema zaustavljanja.

Python interpreter je zapravo Turingov stroj (tehnički Turing potpun jezik). Svaki program koji napišete je opis konačnog automata koji manipulira simbolima na ..traci" (memoriji):

```
def turingov_stroj(traka, program, stanje=0):
    while stanje != "STOP":
        simbol = traka.pročitaj()
        novo_stanje, novi_simbol, smjer = program[stanje][simbol]
        traka.zapiši(novi_simbol)
        traka.pomakni(smjer)
        stanje = novo_stanje
    return traka
```

1.8 Od dedukcije k indukciji

Klasična logika bavi se nužnim zaključcima. Ali većina našeg zaključivanja je probabilistička. David Hume primijetio je „problem indukcije" – iz činjenice da je sunce izašlo svaki dan do sada, ne slijedi nužno da će izaći sutra.

Bayesov teorem daje nam formalni okvir za induktivno zaključivanje:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Gdje je $P(H|E)$ posteriorna vjerojatnost hipoteze nakon evidencije.

U Pythonu:

```
def bayes(prior, likelihood, evidence):
return (likelihood * prior) / evidence
```

```
\textbf{Medicinska dijagnoza}

p_bolest = 0.01  # 1% populacije ima bolest
p_pozitivan_test_ako_bolest = 0.99  # 99% osjetljivost
p_pozitivan_test = 0.05  # 5% ukupno pozitivnih


p_bolest_ako_pozitivan = bayes(
p_bolest,
p_pozitivan_test_ako_bolest,
p_pozitivan_test
)


\textbf{Rezultat: 0.198 ili 19.8%}
```

1.9 Paradoksi i granice

Logika je puna paradoksa koji testiraju naše razumijevanje:

Russellov paradoks: Skup svih skupova koji ne sadrže sebe. Sadrži li sebe?

Paradoks lažljivca: ..Ova rečenica je neistinita." Istinita ili neistinita?

Sorites paradoks: Jedna zrna pijeska nije hrpa. Dodavanje jednog zrna ne čini hrpu. Dakle, nikad nema hrpe?

Ovi paradoksi nisu samo zagonetke – oni otkrivaju fundamentalne limite formalnih sustava i potiču razvoj novih logika (parakontistentne, fuzzy, relevantne).

1.10 Primjena u stvarnom svijetu

Logika kroz kod nije samo akademska vježba. Primjene su svugdje:

Baze podataka koriste logiku predikata (SQL je zapravo logički jezik):

```
SELECT * FROM studenti
WHERE godina > 2 AND prosjek >= 4.0
```

Verifikacija softvera dokazuje korektnost kritičnih sustava:

```
@requires(x >= 0)
@ensures(rezultat >= 0)
```

```
def korijen(x):
    return sqrt(x)
```

Umjetna inteligencija koristi logičko zaključivanje za planiranje i donošenje odluka.

1.11 Putovanje koje slijedi

Ovaj udžbenik vodi vas kroz postupno otkrivanje veze između logike i programiranja. Svaki koncept gradit ćemo od temelja:

1. **Intuicija**: Zašto je koncept važan?
2. **Formalizacija**: Precizna matematička definicija
3. **Implementacija**: Radni Python kod
4. **Eksploracija**: Eksperimenti i varijacije

Ne učite samo *o* logici – prakticirajte logiku kroz kod. Svaka Jupyter bilježnica je laboratorij. Mijenjajte parametre, testirajte granične slučajeve, pokušajte pokvariti kod. Kroz ove eksperimente razvit ćete dublju intuiciju od pukog čitanja.

Zapamtite: u logici, kao i u programiranju, jedini način učenja je činjenjem. Greške su dragocjene – one otkrivaju skrivene pretpostavke i suptilne istine.

Započnimo ovo putovanje kroz svjetove logike, gdje svaki novi koncept otvara vrata dubljeg razumijevanja načina na koji mislimo, zaključujemo i stvaramo.