Matematička logika

Predavanje br. 1 akademske god. 2020/2021

Mladen Vuković

PMF–Matematički odsjek
Sveučilište u Zagrebu

7. listopada 2020.

Konzultacije po dogovoru: **vukovic@math.hr**

Koristit ću web–platformu Merlin za komunikaciju.

Asistentica Lucija Validžić

Sadržaj predavanja:

SADRŽAJ KOLEGIJA

§1. **Logika sudova**

sintaksa: alfabet, formule;

semantika: interpretacija, istinitost, valjanost, ispunjivost;

normalne forme;

testovi valjanosti (vježbe!);

račun sudova;

sistem prirodne dedukcije

intucionistička i modalna propozicionalna logika

§2. Logika prvog reda

sintaksa: alfabet, term, formula, slobodne i vezane varijable;

semantika: struktura, valuacija, inteerpretacija, istinitost, valjanost, ispunjivost;

preneksna normalna forma;

račun logike prvog reda;

generalizirani teorem potpunosti i posljedice;

primjeri teorija prvog reda

LITERATURA

1. M. Vuković, **Matematička logika**, Element, Zagreb, 2009.

2. R. Cori, D. Lascar, **Mathematical Logic** I, II, Oxford University Press, 2000.

3. E. Mendelson, **Introduction to Mathematical Logic**, Chapman&Hall, 1997.

IZVEDBENI PLAN

- Kolokviji:
  - prvi kolokvij (max. 35 bodova)
  - drugi kolokvij (max. 35 bodova);
- Dva kratka testa (max. 10 bodova)
- Završna provjera znanja (max. 20 bodova).

Tijekom semestra pisat će se **dva kratka testa** u zgradi fakulteta.

Kratki test je najavljeni 30 minutni test koji se piše na vježbama.

Točni termini kratkih testova biti će objavljeni tijekom nastave i na web stranici kolegija.

Na svakom kratkom testu može se ostvariti maksimalno 5 bodova.

Za studente koji nisu uspjeli ostvariti barem 35 bodova, i samo za njih, održat će se **popravni kolokvij** u terminu koji će biti oglašen naknadno.

Popravni kolokvij obuhvaća cijelokupno gradivo kolegija (i teorijske zadatke).

Na popravnom kolokviju može se ostvariti maksimalno 60 bodova.

Završna provjera znanja (pismena ili usmena) organizira se za studente koji su ostvarili barem 35 bodova na oba kolokvija, odnosno suma bodova s popravnog kolokvija i kratkih testova, te **trećina bodova ostvarenih na redovnim kolokvijima**, iznosi barem 35.

Za studente koji su negativno ocijenjeni na završnoj provjeri znanja održat će se popravak završne provjere znanja.

Zaključivanje ocjene.

Student mora biti pozitivno ocijenjen na završnoj provjeri znanja, odnosno na popravku završne provjere znanja, kako bi mu se mogla odrediti konačna ocjena.

Na osnovu ocjena iz svih elementa (kolokviji, kratki testovi i završna provjera znanja), nastavnik će odrediti konačnu ocjenu.

Studenti koji su negativno ocijenjeni na završnoj provjeri znanja i na popravku završne provjere znanja moraju ponovno upisati kolegij.

Primjeri (čemu neće biti posvećen ovaj kolegij, odnosno koji su glavni ciljevi)

1. Definicija prekida funkcije u točki.

   Neka je $f : \langle a, b\rangle \to \mathbb{R}$ i $c \in \langle a, b\rangle$.

   Kažemo da je funkcija f neprekidna u točki c ako vrijedi

   $$(\forall \epsilon > 0)(\exists \delta > 0)(\forall x \in \langle a, b\rangle)(|x - c| < \delta \Rightarrow |f(x) - f(c)| < \epsilon)$$

   Kako glasi uvjet da funkcija ima prekid u točki c?

Trebamo negirati uvjet neprekidnosti:

$$\neg(\forall \epsilon > 0)(\exists \delta > 0)(\forall x \in [a, b])(|x - c| < \delta \Rightarrow |f(x) - f(c)| < \epsilon)$$

što je ekvivalentno s

$$(\exists \epsilon > 0)(\forall \delta > 0)(\exists x \in \langle a, b \rangle)\neg(|x - c| < \delta \Rightarrow |f(x) - f(c)| < \epsilon)$$

Kako negirati implikaciju, tj. kako zapisati ekvivalentno tvrdnju

$$\neg(|x - c| < \delta \Rightarrow |f(x) - f(c)| < \epsilon) \ ?$$

2. Nećemo posebno proučavati obrat po kontrapoziciji ili pak De Morganove zakone.

Ne volim baš isticati primjere "iz života" kao što je npr. ovaj "mokri" primjer: "Ako kiša pada onda su ulice mokre".

Čemu uopće treba matematička logika, tj. koji su bili poticaji za razmatranje logičkog zaključivanja?

- Cantorova hipoteza kontinuuma.
  Što je skup? "Skup je pojam koji se ne definira"
- Što je algoritam?
  Hilbertov 10. problem: Odrediti algoritam koji će za svaku diofantsku jednadžbu odgovoriti dali za nju postoji cjelobrojno rješenje.
- Možemo li svaku istinitu matematičku tvrdnju dokazati koristeći unaprijed zadane aksiome i definirana logička pravila?
  NE! Gödelovi teoremi nepotpunosti

Čemu uopće treba matematička logika, tj. koji su bili poticaji za razmatranje logičkog zaključivanja?

- Cantorova hipoteza kontinuuma.
  Što je skup? "Skup je pojam koji se ne definira"

- Što je algoritam?
  Hilbertov 10. problem: Odrediti algoritam koji će za svaku diofantsku jednadžbu odgovoriti dali za nju postoji cjelobrojno rješenje.

- Možemo li svaku istinitu matematičku tvrdnju dokazati koristeći unaprijed zadane aksiome i definirana logička pravila?
  NE! Gödelovi teoremi nepotpunosti

Čemu uopće treba matematička logika, tj. koji su bili poticaji za razmatranje logičkog zaključivanja?

- Cantorova hipoteza kontinuuma.
  Što je skup? "Skup je pojam koji se ne definira"

- Što je algoritam?
  Hilbertov 10. problem: Odrediti algoritam koji će za svaku diofantsku jednadžbu odgovoriti dali za nju postoji cjelobrojno rješenje.

- Možemo li svaku istinitu matematičku tvrdnju dokazati koristeći unaprijed zadane aksiome i definirana logička pravila?
  NE! Gödelovi teoremi nepotpunosti

§ 1. LOGIKA SUDOVA

1.1. Uvod

Primjena logike sudova u računarstvu:

- **Softversko inžinjerstvo** – logika se često koristi za specifikaciju softvera prije njegove izrade
- **Sigurnosne primjene** – "debagiranje" općenito nije dovoljno kako bi se provjerila korektnost nekog programa
- **Dizajn digitalnih spojeva i arhitektura računala** – efikasna minimizacija logičkih spojeva

§ 1. LOGIKA SUDOVA

1.1. Uvod

Primjena logike sudova u računarstvu:

- **Softversko inžinjerstvo** – logika se često koristi za specifikaciju softvera prije njegove izrade
- **Sigurnosne primjene** – "debagiranje" općenito nije dovoljno kako bi se provjerila korektnost nekog programa
- **Dizajn digitalnih spojeva i arhitektura računala** – efikasna minimizacija logičkih spojeva

§ 1. LOGIKA SUDOVA

1.1. Uvod

Primjena logike sudova u računarstvu:

- **Softversko inžinjerstvo** – logika se često koristi za specifikaciju softvera prije njegove izrade
- **Sigurnosne primjene** – "debagiranje" općenito nije dovoljno kako bi se provjerila korektnost nekog programa
- **Dizajn digitalnih spojeva i arhitektura računala** – efikasna minimizacija logičkih spojeva

§ 1. LOGIKA SUDOVA

1.1. Uvod

Primjena logike sudova u računarstvu:

- **Softversko inžinjerstvo** – logika se često koristi za specifikaciju softvera prije njegove izrade
- **Sigurnosne primjene** – "debagiranje" općenito nije dovoljno kako bi se provjerila korektnost nekog programa
- **Dizajn digitalnih spojeva i arhitektura računala** – efikasna minimizacija logičkih spojeva

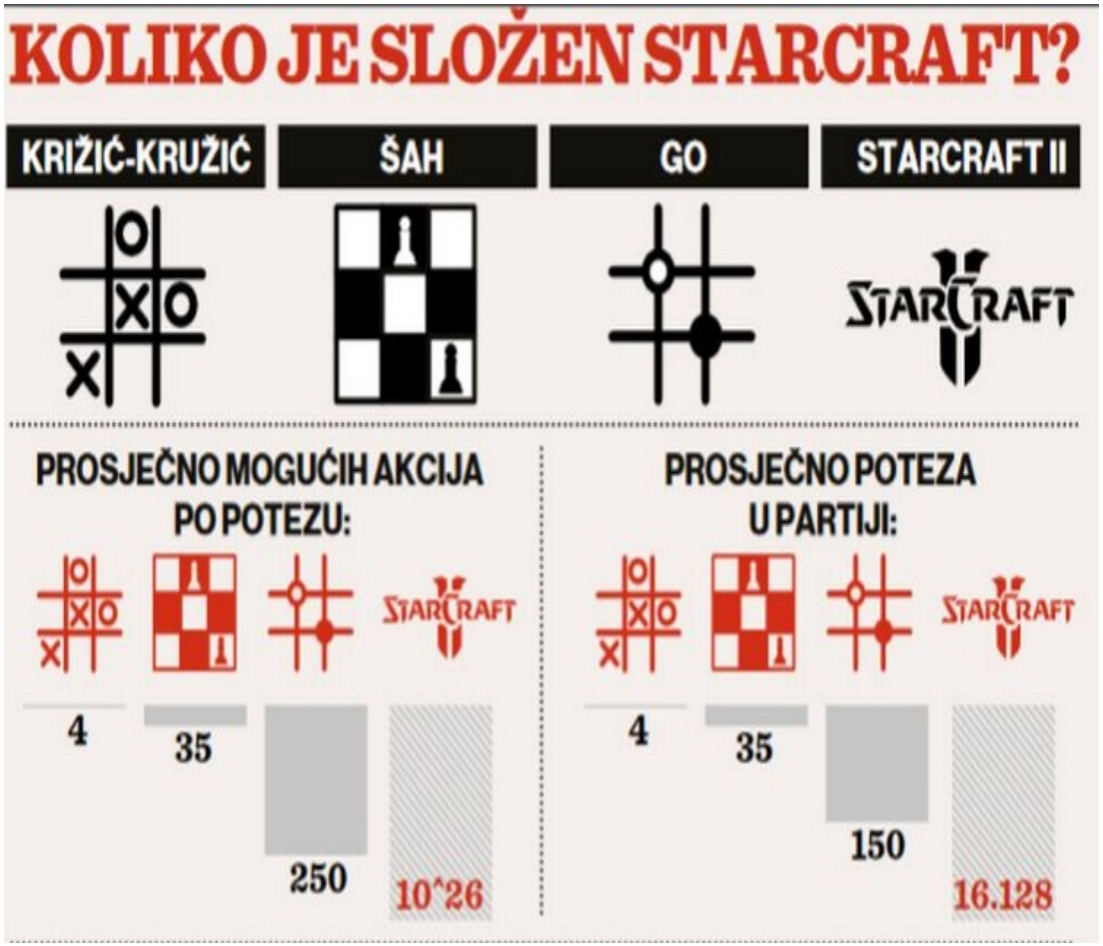
- **Baze podataka** – prilikom postavljanja složenih upita kod pretraživanja neke baze podataka pokazalo se jako važnim prvo pojednostaviti upit
- **Kreiranje** *compilera* – u fazi ispitivanja compilera mora se odrediti je li prevedeni ("compalirani") program takav da je primjena svake varijable i funkcije konzistentna ("zagrebačka priča o compileru za Prolog")
- **Dizajn programskih jezika** – (λ–račun i funkcijsko programiranje)
- **Teorija izračunljivosti** – definiranje apstraktnih strojeva, tj. modela izračunavanja

- **Baze podataka** – prilikom postavljanja složenih upita kod pretraživanja neke baze podataka pokazalo se jako važnim prvo pojednostaviti upit
- **Kreiranje** *compilera* – u fazi ispitivanja compilera mora se odrediti je li prevedeni ("compalirani") program takav da je primjena svake varijable i funkcije konzistentna ("zagrebačka priča o compileru za Prolog")
- **Dizajn programskih jezika** – (λ–račun i funkcijsko programiranje)
- **Teorija izračunljivosti** – definiranje apstraktnih strojeva, tj. modela izračunavanja

- **Baze podataka** – prilikom postavljanja složenih upita kod pretraživanja neke baze podataka pokazalo se jako važnim prvo pojednostaviti upit
- **Kreiranje** *compilera* – u fazi ispitivanja compilera mora se odrediti je li prevedeni ("compalirani") program takav da je primjena svake varijable i funkcije konzistentna ("zagrebačka priča o compileru za Prolog")
- **Dizajn programskih jezika** – (λ–račun i funkcijsko programiranje)
- **Teorija izračunljivosti** – definiranje apstraktnih strojeva, tj. modela izračunavanja

- **Baze podataka** – prilikom postavljanja složenih upita kod pretraživanja neke baze podataka pokazalo se jako važnim prvo pojednostaviti upit
- **Kreiranje** *compilera* – u fazi ispitivanja compilera mora se odrediti je li prevedeni ("compalirani") program takav da je primjena svake varijable i funkcije konzistentna ("zagrebačka priča o compileru za Prolog")
- **Dizajn programskih jezika** – (λ–račun i funkcijsko programiranje)
- **Teorija izračunljivosti** – definiranje apstraktnih strojeva, tj. modela izračunavanja

Umjetna inteligencija



Automatski dokazivači

How Close Are Computers to Automating Mathematical Reasoning?
https://www.quantamagazine.org/how-close-are-computers-to-automating-mathematical-reasoning-20200827/

The Mechanization of Mathematics, Notices AMS, 2018 June/July
https://www.ams.org/journals/notices/201806/rnoti-p681.pdf

Intuitivno, sud je svaka suvisla izjavna rečenica koja je istinita ili lažna, ali ne oboje.

No, to svakako ne može biti definicija suda, jer tada se postavlja pitanje npr. što je rečenica, ili pak što je istinita rečenica.

Pokušat ćemo objasniti pojam suda pomoću nekoliko primjera.

- Rečenica *"Dva plus dva je jednako četiri."* jeste sud i to istinit.
- Rečenica *"Dva plus dva je jednako pet."* jeste sud i to lažan.
- Rečnica *"x plus dva je jednako osam."* nije sud, jer za ovu rečenicu ne možemo reći je li istinita ili lažna, dok nismo rekli koliko je x.

Intuitivno, sud je svaka suvisla izjavna rečenica koja je istinita ili lažna, ali ne oboje.

No, to svakako ne može biti definicija suda, jer tada se postavlja pitanje npr. što je rečenica, ili pak što je istinita rečenica.

Pokušat ćemo objasniti pojam suda pomoću nekoliko primjera.

- Rečenica *"Dva plus dva je jednako četiri."* jeste sud i to istinit.
- Rečenica *"Dva plus dva je jednako pet."* jeste sud i to lažan.
- Rečnica *"x plus dva je jednako osam."* nije sud, jer za ovu rečenicu ne možemo reći je li istinita ili lažna, dok nismo rekli koliko je x.

Intuitivno, sud je svaka suvisla izjavna rečenica koja je istinita ili lažna, ali ne oboje.

No, to svakako ne može biti definicija suda, jer tada se postavlja pitanje npr. što je rečenica, ili pak što je istinita rečenica.

Pokušat ćemo objasniti pojam suda pomoću nekoliko primjera.

- Rečenica *"Dva plus dva je jednako četiri."* jeste sud i to istinit.
- Rečenica *"Dva plus dva je jednako pet."* jeste sud i to lažan.
- Rečnica *"x plus dva je jednako osam."* nije sud, jer za ovu rečenicu ne možemo reći je li istinita ili lažna, dok nismo rekli koliko je x.

- Rečenica *"Ja sada lažem."* nije sud, jer pretpostavimo li da je istinita, onda sam zaista lagao, pa je ono što sam rekao lažno. Obrnuto, pretpostavimo li da je ta rečenica lažna onda nisam lagao, pa je ono što sam rekao istina. Dakle, za ovu rečenicu ne možemo reći ni da je istinita, a ni da je lažna.

- Rečenica *"Koliko je sati?"* nije sud, jer nije izjavna rečenica.

- Rečenica *"Ja sada lažem."* nije sud, jer pretpostavimo li da je istinita, onda sam zaista lagao, pa je ono što sam rekao lažno. Obrnuto, pretpostavimo li da je ta rečenica lažna onda nisam lagao, pa je ono što sam rekao istina. Dakle, za ovu rečenicu ne možemo reći ni da je istinita, a ni da je lažna.

- Rečenica *"Koliko je sati?"* nije sud, jer nije izjavna rečenica.

Prvi i drugi sud iz prethodnog primjera su jednostavnog oblika.

Pomoću veznika *i*, *ili*, *ako ... onda* i *nije* možemo iz jednostavnijih sudova graditi složene.

Na primjer rečenica *"Ako Anđela uči, onda Ivona gleda crtane filmove."* je primjer složenog suda, jer je nastala pomoću veznika *ako ... onda* iz jednostavnih sudova.

U logici sudova proučavamo i **logička zaključivanja**, te određujemo koja su korektna, a koja nisu.

Promotrimo dva primjera. Zaključivanje:

Ako si nabavio ulaznice tada idemo na utakmicu.
Nabavio sam ulaznice.

Idemo na utakmicu.

je naravno primjer korektnog zaključivanje. Formalno zapisano ono je oblika

$$\frac{A \to B \quad A}{B}$$

Nadamo se da se slažete da zaključivanje:

$$\frac{\begin{array}{l}\text{U subotu ću dugo spavati.}\\ \text{Danas nije subota.}\end{array}}{\text{Danas sam rano ustao.}}$$

nije korektno. Formalno ga možemo zapisati u obliku:

$$\frac{\begin{array}{l}A \to B\\ \neg A\end{array}}{\neg B}$$

U ovom poglavlju ćemo definirati što je logička posljedica, tj. koje zaključivanje smatramo korektnim.

1.2. Osnovni pojmovi

Alfabet je proizvoljan neprazan skup.

Svaki element alfabeta nazivamo **simbol** ili **znak.**

Riječ alfabeta je svaki konačan niz danog alfabeta.

Duljina riječi je broj simbola koji dolaze u riječi.

Ako je s A označen neki alfabet tada se skup svih riječi obično označava sa A^*.

Po dogovoru smatramo da skup svih riječi proizvoljnog alfabeta sadrži **praznu riječ**, tj. prazan niz simbola. Praznu riječ obično označavamo s ϵ.

Konkatenacija je binarana operacija na A^*, koja je definirana na sljedeći način:

> *ako su a i b riječi (bolje reći oznake za riječi!) tada kažemo da je riječ ab nastala konkatenacijom riječi a i b.*

Kažemo da je b **podriječ** riječi a ako postoje riječi c i d tako da je riječ a nastala konkatenacijom riječi c, b i d, tj. a je jednaka cbd.

Primjer 1 (Primjeri alfabeta)

Neka je $A_1 = \{\alpha, \beta\}$.

Neke riječi tog alfabeta su npr. $\alpha\alpha\alpha$, $\alpha\beta\alpha\beta\beta\beta$, $\alpha\alpha\beta\beta\alpha\alpha\beta$.

Iz riječi $\alpha\alpha\beta\beta$ *i* $\beta\beta\alpha\beta$ *konkatenacijom dobivamo riječ* $\alpha\alpha\beta\beta\beta\beta\alpha\beta$.

Neka je, zatim, $A_2 = \{+,\ \cdot,\ s,\ 0,\ =\} \cup \{x_n : n \in \mathbb{N}\}$.

Tada su riječi alfabeta A_2 *npr.* $x_1 + x_2 = x_2$, $x_1 \cdot x_4 + 0 = x_5$, *ali i* $++\cdot x_4 ===$.

U sljedećoj propoziciji ističemo činjenicu koju ćemo kasnije često koristiti.

Propozicija 1

Skup svih riječi konačnog ili prebrojivog alfabeta je prebrojiv.

1.3. Sintaksa logike sudova

Definicija 1

Alfabet logike sudova *je unija skupova* A_1, A_2 *i* A_3, *pri čemu je:*

$A_1 = \{P_0, P_1, P_2, \ldots\}$ *prebrojiv skup čije elemente nazivamo* **propozicionalne varijable**

$A_2 = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ **skup logičkih veznika**

$A_3 = \{(,)\}$ **skup pomoćnih simbola (zagrade).**

Uočite da smo u definiciji naveli da alfabet logike sudova sadrži znakove koje nazivamo propozicionalne varijable.

Možete zamišljati da se propozicionalne varijable interpretiraju sudovima, ali to ne mora nužno biti tako.

Jedna interpretacija logike sudova su i npr. elektronički logički sklopovi.

U sljedećoj točki ćemo formalno definirati interpretacije propozicionalnih varijabli.

Logičke veznike redom nazivamo:

$\neg$	**negacija**
$\wedge$	**konjunkcija**
$\vee$	**disjunkcija**
$\rightarrow$	**kondicional**
$\leftrightarrow$	**bikondicional**

Ponekad se definira da alfabet sadrži i znakove $\top$ i $\bot$, koji se nazivaju **logičke konstante** *istina* i *laž*.

Mi ovdje smatramo da ih alfabet ne sadrži.

Definicija 2

Atomarna formula *je svaka propozicionalna varijabla.*

Pojam **formule** *definiramo rekurzivno:*

a) *svaka atomarna formula je formula;*

b) *ako su* A *i* B *formule tada su i riječi*

$$(\neg A), \qquad (A \wedge B), \qquad (A \vee B), \qquad (A \to B) \qquad i \qquad (A \leftrightarrow B)$$

također formule;

c) *riječ alfabeta logike sudova je formula ako je nastala primjenom konačno mnogo koraka uvjeta a) i b).*

Napomena 1

Primijetimo da u prethodnoj definiciji A i B nisu formule već oznake za formule, tj. to nisu simboli jezika već su **meta–simboli**.

Po dogovoru ćemo s velikim slovima (npr. A, B, C, F, G, F_1, F_2, ...) označavati formule.

Za propozicionalne varijable upotrebljavat ćemo oznake P, Q, R, S, ...

Napomena 2 (Dogovor o pisanju zagrada (1))

Način zapisivanja formula obzirom na zagrade, kako smo definirali, naziva se **sistem vanjskih zagrada**.

Zapis formula se može definirati i u **sistemu unutarnjih zagrada** *ili pak u* **poljskoj notaciji**, *tj. bez zagrada.*

Ako su A *i* B *formule, u sistemu unutarnjih zagrada definirali bi da su tada*

$$\neg(A), \qquad (A)\wedge(B), \qquad (A)\vee(B), \qquad (A)\to(B), \qquad i \qquad (A)\leftrightarrow(B)$$

formule.

Napomena 3 (Dogovor o pisanju zagrada (2))

U daljnjem tekstu nećemo se strogo držati zapisivanja formula pomoću zagrada, već ćemo uvesti prioritet logičkih veznika.

Najveći **prioritet** *ima negacija, zatim veznici* $\wedge$ *i* $\vee$, *a najmanji prioritet (ali isti) imaju veznici* $\rightarrow$ *i* $\leftrightarrow$.

No, to ne znači da ćemo se potpuno odreći zagrada prilikom zapisivanja formula.

U nekim situacijama ćemo pisati zagrade kako bi istaknuli prioritet nekog veznika.

Primjer 2

Zapis formule:

- *sistem vanjskih zagrada:* $(((\neg P) \wedge Q) \to R)$
- *sistem unutarnjih zagrada:* $((\neg(P)) \wedge (Q)) \to (R)$,
- *obično ćemo zapisivati kao:* $(\neg P \wedge Q) \to R$.

Kažemo da je formula B **potformula** formule A ako je riječ B podriječ od A.

Neka je A formula te neka je $\{P_1, \ldots, P_n\}$ skup svih propozicionalnih varijabli koje se pojavljuju u A. To kratko označavamo sa $A(P_1, \ldots P_n)$.