**Predmet: "Vještačka inteligencija"**
**Laboratorijska vježba 8: Ekspertni sistemi (2/2)**
**Odgovorna nastavnica: Vanr. prof. dr Amila Akagić**


# Sadržaj vježbe:



# 1 Cilj vježbe

Kroz ovu vježbu studenti će se upoznati sa pojmom pravila lančanja unazad, načinom pisanja tih pravila (nakon što su na prethodnoj vježbi objašnjena pravila lančanja unaprijed), kao i sa pojmom i načinom kreiranja baza pitanja, kao jednim od tri dijela baze znanja u PyKE-u.

# 2 PyKE (2. dio)

## 2.1 Baza pravila (pravila lančanja unazad)

Na prethodnoj laboratorijskoj vježbi je rečeno da pravila u PyKE-u (koja su pohranjena u datoteci sa ekstenzijom `.krb`) mogu biti pravila lančanja unaprijed i pravila lančanja unazad.

Pravila lančanja unazad koriste ključne riječi `use` i `when` umjesto `THEN` i `IF`, respektivno, i funkcionišu na sljedeći način:

1. PyKE prvo pronalazi ono pravilo čiji `THEN` dio odgovora postavljenom cilju (koji se postavlja u Python programskom jeziku).
2. Nakon što pronađe pravilo, PyKE provjerava da li su ispunjeni svi uslovi postavljeni u `IF` dijelu tog pravila (`when` dio), pri čemu:
   - uslovi mogu biti *primitivni*, odnosno, uslovi čije se ispunjenje može provjeriti direktno iz baze činjenica;
   - uslovi mogu biti i kompleksni, u smislu da je potrebno provjeriti ispunjenje nekog drugog pravila u bazi pravila (koje se koristi i kao uslov za pravilo koje se trenutno provjerava), odnosno, može doći do grananja među pravilima;
3. Ukoliko su sva pravila ispunjena, originalni (postavljeni) cilj je ispunjen i dokazan. U suprotnom, PyKE traži neko drugo pravilo na način opisan u koraku 1, te ukoliko postoji, ponavljaju se koraci 2 i 3.

U primjeru ekspertnog sistema za porodične relacije može se naći baza pravila (pravila lančanja unazad) pohranjena u datoteku `bc_example.krb`. Primjer jednog pravila (iz te baze) prikazan je u nastavku:

```
father_son
   use child_parent($child, $father, father, son)
   when
      family.son_of($child, $father, $mother)
```

Za razliku od pravila lančanja unaprijed, kod pravila lančanja unazad prvo se navodi `THEN` dio, za koji se koristi ključna riječ `use` (u ovom slučaju to je `child_parent($child, $father, father, son)`, a onda se nakon

toga navodi jedna ili više činjenica (iako je u ovom slučaju samo jedna) koje trebaju biti ispunjene da bi činjenica navedena u `THEN` dijelu bila ispunjena, odnosno, navodi se `IF` dio pravila. Dakle, može se vidjeti da je redoslijed `IF` i `THEN` dijela obrnut u odnosu na pravila lančanja unaprijed, te da se koriste druge ključne riječi - `use` i `when`.

Način na koji funkcioniše dokazivanje postavljenog cilja korištenjem pravila lančanja unazad može se prikazati i na jednom konkretnom primjeru. Pretpostavimo da ekspertni sistem treba da odgovori na pitanje ko je otac od Bruce-a. U tom slučaju, cilj koji treba dokazati (a koji se definiše u programskom jeziku Python, na osnovu pravila iz baze pravila) treba postaviti na način `child_parent(Bruce, $father, father, son)`. Sada, kada se pokrene ekspertni sistem, prema nizu koraka predstavljenom u prethodnom dijelu, PyKE pronalazi (prvo) pravilo čiji `THEN` (odnosno `use`) dio odgovara postavljenom cilju. Prvo takvo pravilo je definisano na način prikazan ispod:

```
father_son
   use child_parent($child, $father, father, son)
   when
      family.son_of($child, $father, $mother)
```

Nakon što je detektovano pravilo od interesa, cilj je provjeriti da li su ispunjeni uslovi ovog pravila, odnosno, posmatra se `when` dio ovog pravila. U ovom slučaju, `when` dio je definisan samo jednom činjenicom čije ispunjenje treba provjeriti, pri čemu se radi o *primitivnoj* činjenici, odnosno, činjenici koja se provjerava kroz bazu činjenica (`family.kfb`) i nema daljeg grananja unutar baze pravila (s obzirom da se i među uslovima koje treba provjeriti može naći neka od činjenica koja se aktivira kroz bazu pravila). S obzirom na definisano pravilo i na postavljeni cilj, jasno je da je vrijednost prvog parametra (`$child`) jednaka `Bruce`, odnosno, varijabla `child` se postavlja na `Bruce`, pa pravilo poprima sljedeći oblik:

```
father_son
   use child_parent(Bruce, $father, father, son)
   when
      family.son_of(Bruce, $father, $mother)
```

Da bi se provjerilo da li je `when` dio ispunjen, potrebno je konsultovati bazu činjenica, odnosno, datoteku `family.kfb`, s obzirom da se radi o činjenici koju je moguće pronaći u toj datoteci. Potrebno je pronaći činjenicu oblika `son_of(Bruce, $father, $mother)`, odnosno, činjenicu u kojoj je vrijednost prvog parametra postavljena na `Bruce`, dok vrijednost druga dva parametra može biti jednaka bilo kojoj vrijednosti. U bazi činjenica je jedna takva činjenica - `son_of(bruce, thomas, norma)`, na osnovu čega se zaključuje da vrijedi `father = thomas`, odnosno, ekspertni sistem će odgovoriti da je Thomas otac od Bruce-a.

Neka sada ekspertni sistem treba odgovoriti na pitanje da li su Bruce i Jim prvi rođaci. U tom slučaju, cilj je potrebno definisati na sljedeći način: `cousins(Bruce, Jim, 1)`. Sada je ponovo potrebno pronaći pravilo čiji `THEN` dio odgovara postavljenom cilju i prvo takvo pravilo ima sljedeći oblik:

```
first_cousins
   use cousins($cousin1, $cousin2, 1)
   when
      child_parent($cousin1, $sibling1, $_, $_)
      siblings($sibling1, $sibling2, $_, $_)
      child_parent($cousin2, $sibling2, $_, $_)
```

Iz priloženog se može vidjeti da se radi o pravilu koje sadrži nekoliko činjenica koje treba dokazati (a ne samo jednu, za razliku od do sada prikazanih pravila), pri čemu su sve činjenice *kompleksne*, odnosno, nijedna od činjenica se ne može provjeriti gledajući direktno u bazu činjenica, nego je potrebno provjeriti ispunjenost ovih činjenica kroz neka druga pravila u bazi pravila. Prije svega, objasnimo da se, prema ovom pravilu, prvim rođacima smatraju dvije osobe čiji roditelji su u odnosu brat/sestra. Također, na osnovu definisanog cilja, vrijednost varijable `cousin1` je `Bruce`, dok je vrijednost varijable `cousin2` `Jim`. Prvo pravilo čiju ispunjenost je potrebno provjeriti je `child_parent(Bruce, $sibling1, $_, $_)`, po već utvrđenom algoritmu, odnosno, na način da se prvo pronalazi pravilo koje ispunjava ovu formu pravila. Takvih je nekoliko pravila u bazi pravila (četvero) i potrebno je ispitati svako od njih (proces dokazivanja `child_parent` je već demonstriran kroz prethodni primjer) i nakon provjere tih pravila doći će se do zaključka da vrijednost varijable `sibling1` može uzimati dvije vrijednosti - `thomas` i `norma`.

Nakon što je provjerena prva činjenica, prelazimo na drugu činjenicu, koja je također kompleksnog tipa, a to je činjenica koja treba pronaći sve osobe koje su brat/sestra već pronađenim osobama (varijabli `sibling1` - `thomas` i

norma). U bazi pravila se također nalazi nekoliko pravila koja ispunjavaju traženu formu (siblings($sibling1, $sibling2, $_, $_)) i na ovom mjestu neće biti prikazan detaljan proces dokazivanja ove činjenice (studentima se ostavlja da to urade sami), ali se na kraju ispostavi da su joyce i phyllis u odnosu brat/sestra sa thomas-om, dok su john_w, bill i chuck_w u odnosu brat/sestra sa norma-om, pa tako vrijednost varijable sibling2 može uzimati pet vrijednosti - joyce, phyllis, john_w, bill i chuck_w.

Sada još ostaje da se provjeri treća činjenica, odnosno, gdje je potrebno provjeriti da li su cousin2 (odnosno jim) i sibling2 (odnosno joyce, phyllis, john_w, bill i chuck_w) u odnosu dijete/roditelj. Detaljnijim ispitivanjem dolazi se do zaključka da su jim i bill u takvom odnosu (bill je otac od jim-a), što znači da je i treća činjenica dokazana, što dalje znači da je postavljeni cilj dokazan, odnosno, Bruce i Jim jesu prvi rođaci (s obzirom da su Bill, otac od Jim-a, i Norma, majka od Bruce-a, brat i sestra).

Još jedna od razlika između pravila lančanja unaprijed i pravila lančanja unazad je u tome što kod pravila lančanja unazad ne postoji pojam aktiviranja pravila (koji smo imali kod pravila lančanja unaprijed). Posljedica toga se vidi u načinu definisanja *driver*-a. Naime, ukoliko otvorite driver.py datoteku za ovaj ekspertni sistem i pogledate bc_test funkciju (funkcija koja testira rad eskpertnog sistema za pravilima lančanja unazad), može se vidjeti da se cilj definiše na način da se, umjesto naziva baze činjenica, specificira naziv baze pravila (bc_example), te da se cilj ne definiše korištenjem comppile funkcije, nego se prosljeđuje kao prvi parametar funkcije prove_goal, koja se pozive nad kreiranim engine-om. Način pokretanja ekspertnog sistema koji radi sa pravilima lančanja unazad je identičan načinu objašnjenom na prethodnoj vježbi - potrebno je da se *import*-uje *driver* te da se nad tim *driver*-om pozove bc_test funkcija.

## 2.2 Baza pitanja

Na prethodnoj vježbi je rečeno da se u PyKE-u baza znanja definiše kroz bazu činjenica, bazu pravila i bazu pitanja, pri čemu su prve dvije stavke detaljnije objašnjene na prethodnoj i dijelom na ovoj vježbi, dok će baza pitanja biti objašnjena u ovom dijelu vježbe. Baza pitanja je pohranjena kao .kqb datoteka (pri čemu skraćenica .kqb dolazi od skraćenice za *Knowledge Question Base*). U ovoj bazi se pohranjuju pitanja koja se prikazuju kranjem korisniku na koja on odgovora i uz pomoć kojih korisnik dolazi do konačnog odgovora ekspertnog sistema. Pitanja mogu biti različitog tipa: *Yes/No* pitanja, *True/False* pitanja ili *multiple choice* pitanja.

U primjeru ekspertnog sistema, koji je do sad korišten na vježbama za demonstraciju osnovnih principa, ne postoji definisana baza pitanja (jasno je da baza pitanja nije obavezna za definisati da bi ekspertni sistem radio), tako da se, za potrebe demonstriranja baze pitanja, studenti upućuju na drugi primjer - primjer ekspertnog sistema za učenje PyKE-a, smještenog u folder learn_pyke, koji se također nalazi u examples folderu (preuzetom na prethodnoj vježbi). Unutar learn_pyke foldera nalazi se datoteka questions.kqb, unutar kojeg su definisana pitanja. Primjer jednog pitanja za korisnika prikazan je ispod:

```
knows_prolog($ans)
   Do you have some familiarity with the programming language prolog?
   --
   $ans = yn
```

Dakle, prvo se navodi naziv pitanja, zatim tekst pitanja, i na kraju, korištenjem ključne riječi $ans navodi se format odgovora na pitanja (u ovom slučaju se radi o pitanju *Yes/No* tipa, koji se specificira korištenjem skraćenice yn, te se od korisnika da, prilikom pokretanja ekspertnog sistema, unese y ili n). Kako je već rečeno, moguće je koristiti i *multiple choice* pitanja, što je i demonstrirano na primjeru ispod (također jedno od pitanja iz baze pitanja za ekspertni sistem za učenje PyKE-a):

```
tuple_pattern_syntax($ans)
   What is the syntax for a tuple pattern?
   --
   $ans = select_1
      1:  A series of patterns enclosed in "tuple(" ...  ")".
      2.  A series of patterns enclosed in parenthesis.
```

Iz priloženog primjera se može vidjeti da se vrijednost $ans parametra postavlja na select_1, čime se signalizira da je potrebno odabrati jedan od ponuđenih odgovora (u ovom slučaju, dva odgovora su ponuđena i svaki od njih je

numerisan).

Sada još ostaje da vidimo na koji način se baza pitanja povezuje sa ostalim bazama, i generalno, sa ekspertnim sistemom. Baza pitanja se povezuje sa bazom pravila i to na isti način na koji se povezuje baza činjenica sa bazom pravila. Naime, u folderu `learn_pyke` se nalazi i baza pravila, pohranjena u `pattern_matching.krb`, gdje se može i vidjeti na koji način se vrši referenciranje na pitanja iz baze pitanja. Jednostavan primjer (iz te datoteke) je prikazan ispod:

```
prior_knowledge1
   use prior_knowledge()
   when
       questions.knows_prolog(True)
```

Radi se o pravilu lančanja unazad (s obzirom da se koriste `use` i `then` ključne riječi), te se može vidjeti da se u ovom pravilu vrši referenciranje na pitanje `knows_prolog` (koje je već ranije spomenuto), pri čemu se to referenciranje vrši na isti onaj način na koji se vrši i referenciranje na neku od činjenica (pohranjenih u bazi činjenica) - navođenjem imena baze (datoteke) sa pitanjima (u ovom slučaju je to `questions`), nakon čega se navodi naziv pitanja. Dakle, da bi ovaj ekspertni sistem ispravno radio potrebno da je pitanja budu definisana u datoteci `questions.kqb`. Također, može se primijetiti da se unutar `learn_pyke` foldera ne nalazi baza činjenica (nema datoteke sa ekstenzijom `.kfb`), tako da se može zaključiti da definisanje baze činjenica nije neophodno za ispravno funckionisanje ekspertnog sistema u PyKE-u. Korisno je spomenuti i način na koji se vrši referenciranje na pitanja u slučaju *multiple chocie* pitanja. U nastavku je prikazano jedno takvo pravilo (također iz baze pravila za ekspertni sistem za učenje PyKE-a):

```
taught_pattern_matching_3_5_6_9_15
   use taught_pattern_matching()
   when
      questions.pat_master($ans)
      check $ans in (3, 5, 6, 9, 15)
      knows_tuple_patterns()
```

Ovdje se vrši referenciranje na pitanje `pat_master`, koje ima nekoliko ponuđenih odgovora, a da bi ovo pravilo bilo zadovoljeno, potrebno da je, između ostalog, i da za odgovor na pitanje `pat_master` bude ponuđena jedna od 5 navedenih opcija - odgovor sa rednim brojem 3 ili 5 ili 6 ili 9 ili 15. Koristi se ključna kriječ `check` nakon čega se navodi naziv varijable - `$ans`, te se nakon toga u zagradi navode redni brojevi odgovora koji su dozvoljeni. Također, ukoliko postoji samo jedan odgovor koji je dozvoljen (npr. samo odgovor sa rednim brojem 3), onda je potrebno koristiti sljedeću sintaksu da bi sistem ispravno radio: `check $ans in (3,)`.

Studentima se preporučuje da pokrenu i ovaj ekspertni sistem na način na koji su pokretali i ekspertni sistem za porodične relacije (prvo *import*-ovati *driver* i onda pozvati funkciju koja vrši testiranje, a u ovom slučaju je to funkcija `run`).

# 3 Zadaci za rad u laboratoriji

Zadaci u nastavku su predviđeni za samostalni rad studenata.

## Zadatak 1 - Ekspertni sistem za porodične relacije

U primjeru na kojem je demonstrirano kako ekspertni sistem dolazi do odgovora da li su Bruce i Jim prvi rođaci nisu prikazani detalji prilikom dokazivanja svake pojedinačne činjenice (ukupno su 3 takve činjenice), nego su prikazani samo konačni odgovori. Za svaku od činjenica, detaljno prikazati kako se došlo do odgovora koji su navedeni.

## Zadatak 2 - Ekspertni sistem za vrijeme

Prepraviti rješenje zadatka 2 sa laboratorijske vježbe 7 na način da se u bazi pravila, umjesto pravila lančanja unaprijed, koriste isključivo pravila lančanja unazad. Eskpertni sistem bi trebao i dalje da odgovara na isto pitanje i to u formatu koji je naveden u postavci zadatka 2 na prethodnoj vježbi.

## Zadatak 3 - Ekspertni sistem za vrijeme sa pitanjima

Prepraviti rješenje prethodnog zadatka na način da se umjesto (hardkodirane) baze činjenica koristi baza pitanja (baza činjenica se uopće ne treba koristiti). Baza pitanja treba imati samo dva pitanja - *Da li pada kiša?* i *Da li je vjetrovito?*, te da se u zavisnosti od odgovora korisnika ispiše odgovarajuća poruka o tome šta bi korisnik trebao ponijeti sa sobom. Eskpertni sistem bi trebao i dalje da odgovara na isto pitanje i to u formatu koji je naveden u postavci zadatka 2 na prethodnoj vježbi.

## Zadatak 4 - Ekspertni sistem za vrijeme u doba korone sa pitanjima

Pretpostaviti da se javio još jedan ekspert koji je savjetovao da, ukoliko je poznato da se desila poplava, onda ekspertni sistem treba da kaže korisniku da sa sobom ponese i gumene čizme. Jedan drugi ekspert je savjetovao da, ukoliko je poznao da vani vlada pandemija, onda ekspertni sistem treba da kaže korisniku da sa sobom ponese i masku. Potrebno je prepraviti rješenje zadatka 3 na način da ekspertni sistem uzme u obzir i ove slučajeve.

Mišljenja ova dva eksperta potrebno je uvažiti kroz definisanje jednog *multiple choice* pitanja u bazi pitanja, koje treba da odgovori na pitanje šta se trenutno vani dešava - polava ili pandemija (za potrebe ovog zadatka pretpostaviti da su ove dvije pojave međusobno isključive), ili ni jedno ni drugo.

Prepraviti rješenje prethodnog zadatka na način da se uvaži mišljenje i ovih eksperata. Eskpertni sistem sada treba da ispiše dvije poruke - prva poruka koja zavisi od toga da li je vani kišovito i/ili vjetrovito (ono što je sistem i do sada radio), a druga poruka koja zavisi od toga da li je vani poplava ili pandemija.